UNIFRAME RESOURCE DISCOVERY SERVICE MONITORING AND MANAGEMENT SYSTEM

TR-CIS-0411-05

Srikanth Reddy April 07, 2005

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **APR 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **Uniframe Resource Discovery Service Monitoring and Managment System**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Indiana University/Purdue University,Department of Computer and Information Sciences,Indianapolis,IN,46202**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**193**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

UNIFRAME RESOURCE DISCOVERY SERVICE MONITORING AND MANAGEMANT SYSTEM

A Technical Report

Report Number

TR-CIS-0411-05

Submitted to the Faculty

Of Purdue University

By

Srikanth Reddy

In Partial Fulfillment of the

Requirements for the Degree

Of

Master of Science

April 2005

To Amma & Nana.

ACKNOWLEDGMENTS

The graduate education at Department of Computer and Information Science at Indiana University – Purdue University –Indianapolis has been an excellent experience. I would like to take the opportunity to express my gratitude to all those who made my graduate study, a memorable experience and making this project possible.

I would like to thank my advisor, Dr. Rajeev R. Raje for his continuous guidance and support during the course of my project. His continuous feedback and encouragement helped me laying a strong foundation for the project. I would also thank him for his encouragement in learning and exploring new concepts.

I would like to thank Dr. Andrew M. Olson for being on my advisory committee and for his valuable advice in successfully completing the project. His meticulous efforts to review my report and work are greatly appreciated.

I would like to thank Dr. Snehasis Mukhopadhyay for being on my advisory committee and providing valuable inputs towards the project.

I would like to thank the U.S. Department of Defense and the U.S. Office of Naval Research for supporting this research under the award number N00014-01-1-0746.

Many thanks to the faculty and staff of the Department of Computer and Information Science for their cooperation.

I would like to thank my family for their support and for providing an opportunity to study at IUPUI. Last but not the least; I would like to thank all my teammates of UniFrame project and friends for their encouragement and assistance throughout the course of my study.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF TERMS

Acronym	**Term**
AM	Adapter Manager
AMIDST	Application of Middleware for Services in Telimatics
API	Application Programming Interface
AR	Active Registry
CBSD	Component-Based Software Development
CORBA	Common Object Request Broker Architecture
DCS	Distributed Computing System
DSM	Domain Security Manager
FRIENDS	FRamework Integrated ENgineering and Deployment of Services
GDM	Generative Domain Model
GGF	Global Grid Forum
GMA	Grid Monitoring Architecture
gmetad	Ganglia Meta Daemon
gmond	Ganglia Monitoring Daemon
GIIS	Grid Index Information Service
GRIS	Grid Resource Information Service
GUI	Graphical User Interface
HH	Headhunter
HTTP	Hypertext Transfer Protocol
ICB	Internet Component Broker
IDL	Interface Definition Language
J2EE	Java^{TM} 2 Platform Enterprise Edition
J2SE	Java^{TM} 2 Platform, Standard Edition
JDBC	Java DataBase Connectivity
JRMP	Java Remote Method Protocol

JSP	Java Server Pages ^{TM}
LDAP	Lightweight Directory Access Protocol
LM	Link Manager
MDS	Monitoring and Discovery Service
MR	Meta-Repository
MVC	Model-View-Controller
.NET	Microsoft .NET
OMG	Object Management Group
QM	Query Manager
QoS	Quality of Service
RDBMS	Relational Database Management System
R-GMA	Relational Grid Monitoring Architecture
RMI	Remote Method Invocation
RMI-IIOP	Remote Method Invocation over Internet Inter ORB Protocol
S-expressions	Symbolic expressions
SI	System Integrator
TCP	Transmission Control Protocol
UA	UniFrame Approach
UI	User Interface
UMM	Unified Meta-component Model
URDS	UniFrame Resource Discovery Service
URDSMMS	UniFrame Resource Discovery Service Monitoring and Management System
WSDL	Web Services Description Language
XML	EXtensible Mark-up Language

ABSTRACT

Reddy, Srikanth, M.S., Purdue University, May 2005. URDS – Monitoring and Management System. Major Professor: Dr. Rajeev R. Raje.

Application development and deployment is moving from centralized sequential systems towards parallel and Distributed Computing Systems (DCS), and there has been a growing trend of developing software systems using a Component-based Software Development (CBSD) approach. UniFrame is one such approach providing a seamless framework for constructing a DCS by semi-automatic integration of heterogeneous distributed software components. UniFrame Resource Discovery Service (URDS) is the constituent of UniFrame, responsible for discovering software components. Since URDS consists of many concurrent processes, understanding its execution behavior is a major challenge. Such a discovery service requires a framework that provides information about the execution behavior of the software system. The research of this project designs and implements such a monitoring framework, UniFrame Resource Discovery Service – Monitoring and Management System (URDSMMS) for the URDS. The monitoring framework includes the tasks of collecting dynamic behavioral data, interpretation of this data, and display of the processed information. A user interface was also developed for easy operation of the monitoring system. To ascertain the benefits of this monitoring system, an experimental analysis using the prototype has also been performed.

1. INTRODUCTION

The rapid advancement in the processor and networking technologies and the need for a greater computing power has led to the development of Distributed Computing Systems (DCSs). A DCS is defined as sharing of hardware or software components located at different computers connected over a network which communicate and coordinate by passing messages [COU01]. Typically components in a DCS are characterized by their geographical isolation and non-sharing of physical memory. As a result the integration of these disparate components is a challenging task in the construction of a DCS.

Developing software from already available software components has been a growing trend in building software systems. This is achieved by using Component-Based Software Development (CBSD), which aims in developing software systems by assimilating previously available software components. This CBSD approach can be applied in the construction of a DCS, where the components have public interfaces and private implementations, and are independently created and deployed over a network. These deployed components can be heterogeneous, which makes the construction of a DCS a complex task.

An important step in building a DCS using the component-based approach is the discovery of heterogeneous software components. These components should regularly discover one another, offer and utilize services, and agree on the cost and quality of services. Such a scheme should provide for a scalable solution and hide the underlying heterogeneity [RAJ01] of a DCS. UniFrame [RAJ00] project provides a solution for this problem.

1.1. UniFrame

The UniFrame Approach (UA) [RAJ01] provides a comprehensive framework that allows a seamless interoperation of heterogeneous and distributed software components.

This framework incorporates the following concepts; (i) a meta-component model (the Unified Meta Model - UMM) [RAJ00], with a hierarchical setup for indicating the contracts and constraints of the components, (ii) an integration of Quality of Service (QoS) at both individual component level and distributed system level, (iii) a mechanism for the validation and assurance of QoS aspects, based on event grammars, and (iv) generative rules, along with their formal specifications, for integrating an ensemble out of available components.

The UA specifies a framework for component developers to create, test and validate components from the Quality of Service (QoS) perspective. The developed components are then deployed on the network and are available for system integrators to select and semi-automatically generate a software solution for the DCS under consideration. The framework also incorporates a resource discovery service called UniFrame Resource Discovery Service (URDS) [SIR03]. The URDS is responsible for discovering the software components dynamically and hence, providing the System Integrators a directory-style access to services. Chapter 2 presents an overview of URDS architecture.

1.2. Problem Statement and Motivation

The URDS, the discovery service entity of UniFrame, contains many concurrent processes for carrying out components discovery. The challenging issue in this discovery service is to control and coordinate different processes. To manage such processes, there is a need for a monitoring system. While numerous monitoring systems have been developed for different DCS, such as Hawkeye [HAW], Monitoring and Discovery Service (MDS) [MDS2], Ganglia [MAS04], they are not specifically developed for software resource discovery services. Since these system deals with hardware related resources, they do not focus on software resources. This research project details a monitoring system that can be used to dynamically monitor and administer the various processes and entities encompassing the URDS.

The motivation for creating a monitoring service for URDS is as follows: In URDS, the component selection is based on their computational, co-operational, auxiliary attributes and QoS metrics [SIR03]. To do this process, URDS consists of numerous entities which communicate among themselves. Because of this communication, large numbers of inter-process and intra-process messages are issued. To control and coordinate these, URDS must be monitored throughout its execution so that the system behavior can be observed based on the state changes and failures. Observing this behavior would help the system administrator in determining the overhead of the URDS and hence could decide on deploying new entities or not.

Monitoring would assist the system administrator with information about each entity in URDS and its activities on the remote/local machines. Monitoring would also aid in providing details such as the response time, utilization of different entities of URDS, etc. This would help to improve the throughput of the system and hence the overall performance of URDS. Presently, URDS lacks such a monitoring system. Consequently, a system administrator is forced to depend on manual supervision for detecting faults or doing any other kind of monitoring. A monitoring system is essential for abstracting the details of the underlying system, making it possible even for non-expert users to oversee the functioning of the system and take preventive measures, and hence acting as a controlling authority. The UniFrame Resource Discovery Service - Monitoring and Management System (URDSMMS) is designed to provide these facilities for monitoring and administering the system.

1.3. Objectives : Statement of Goals

The project entails the development of a URDSMMS, through which system administrators can control and coordinate different activities in URDS, and system integrators submit queries and select components to compose distributed software systems.

The specific objectives of this project include:

- To analyze the behavior of entities in URDS.
- To develop a system monitoring prototype for URDS, encompassing system administrator and system integrator views.
- To provide facilities for management of URDS, by offering services such as, initiation and termination of different entities in URDS.

URDSMMS aims at providing the above functionality via a graphical interface. The system continuously provides the details of various entities of through this graphical interface.

1.4. Contributions

The contributions of this project are:

- It proposes a framework for dynamic monitoring and management of a resource discovery service.
- It develops a monitoring and management interface, incorporating two views; system administrator and system integrator, for the operation of URDS.
- It implements time-driven and event-driven approaches for monitoring components of a discovery service.

1.5. Organization of this Report

The report is organized into six chapters. An introduction, along with the problem definition and motivation, objectives, and the contributions are provided in this chapter. Chapter 2 discusses the background and related work of this report. Chapter 3 describes the existing monitoring systems, and features that can be implemented for a monitoring and management system of URDS. Chapter 4 gives the design and implementation details of the different entities that form the URDSMMS and also the design of interface for the monitoring system. The experimentation of URDSMMS and analysis about, how the features are helpful in analyzing the system is discussed in Chapter 5. Chapter 6 provides the conclusion of this research work, the possible future enhancements, and the summary of this report.

2. BACKGROUND AND RELATED WORK

This chapter presents an overview of the UniFrame, which forms a basis for URDSMMS. The chapter also provides an overview of the other related monitoring systems and discusses some of the approaches to monitor software components.

2.1. UniFrame

As indicated in the first chapter, the objective of UniFrame is to provide a framework for a seamless integration of heterogeneous and distributed software components. The UniFrame consists of the Unified Meta-component Model (UMM) [RAJ00] and the UniFrame Approach (UA) [RAJ01]. The UA is a component-based software development approach based on UMM for creating a DCS out of available heterogeneous distributed software components. A brief description of UMM is presented in sub-section 2.1.1 and a description about UA is presented in 2.1.2.

2.1.1. Unified Meta-Component Model (UMM)

The Unified Meta-component Model (UMM) [RAJ00] is the fundamental concept behind the UniFrame project. It unifies the existing and emerging distributed component models under a common meta-model. The UMM helps in allowing resource discovery, interoperability and collaboration among the heterogeneous components. The UMM consists of three parts and they are as follows:

a) *Components:* In UniFrame, components are autonomous entities; they adhere to different distributed component models, and maintain a state, identity and behavior. These components have well-defined (public) interfaces and private implementations. Additionally each component in UMM have three aspects namely (i) Computational Aspect, (ii) Cooperative Aspect, and (iii) Auxiliary Aspect.

The *computational aspect* reflects the task(s) carried out by each component which in-turn depends on the objectives of the task, techniques used to accomplish the objective and a specification about the functionality of the component. The UMM indicates the computational aspect using a mixed approach in which the informal text is used to provide book keeping information about the component, and a precise formal part for the description of computation, its associated contracts and the level of service offered by the component. The *cooperative aspect* indicates the interaction of the components. It indicates the other components that can collaborate with the component under consideration. The *auxiliary aspect* deals with issues such as, the mobility, security and fault tolerance of the component. [RAJ00] describes in detail about the above three aspects.

b) *Services and Service Guarantees:* A service as defined in [RAJ00] could be an intensive computational effort or an access to its underlying resources. In UniFrame, components that offer services should provide certain Quality of Service (QoS) guarantees, so that those components can be selected for the construction of a DCS. The QoS offered by each component depends on, the computation performed, the algorithm used, the expected computational effort and resources required, the cost of each service, and the dynamics of supply and demand [RAJ00]. In UMM it is necessary to specify the QoS that the component can offer in terms of the QoS parameters listed in the QoS catalog [BRA02].

c) *Infrastructure:* In UMM, the infrastructure which is necessary for the discovery of components is provided by UniFrame Resource Discovery Service (URDS). The deployed components can belong to different component models and their discovery is based on certain functional and non-functional constraints. The URDS infrastructure is primarily made up of Headhunters and Internet Component Broker [RAJ01]. Some of the aspects of URDS will be further discussed in detail in section 2.1.3.

2.1.2. UniFrame Approach (UA)

The UA [RAJ01] attempts in providing a framework for the integration of distributed heterogeneous software components. The UA has two levels for the generation of a DCS: a) the component development and deployment (component level), and b) an automatic generation of system using components and validation of system QoS (system level). The component level allows component developers to create, test and verify components from the point of view of QoS requirements and deploy them on a network. The system level allows system integrators/component assemblers to select and generate a software solution for a DCS under consideration by an automatic or semi-automatic generation process.

The UniFrame Approach uses a Generative Programming [CZA00] paradigm, to generate a system from components. Therefore, this process has an underlying assumption that a DCS is built around a knowledgebase which supports the component assembly. Here, the components are created for a specific application domain, based on an accepted and a standardized knowledgebase. The UniFrame knowledgebase (as shown in Figure 2.1) is assumed to be created by domain experts. The knowledgebase consists of three parts: general information, which includes a description for the modeled domain; a problem space, used by an application programmer to specify the needs; and a solution space, which contains various models including configuration knowledge to provide solutions for a DCS family. The UniFrame provides a specification mechanism called UMM Specification. The component developer uses this specification to develop components for a particular domain. Subsequently, the developed component along with its specification is deployed over a network. A sample UMM specification template is provided in Table 2.1.

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for DocumentValidationServer in the banking
domain example -->

<UMM_ConcreteComponent>
    <ComponentName> UserValidationServer </ComponentName>
    <ComponentSubcase> UserValidationServerCase1 </ComponentSubcase>
    <DomainName> Document </DomainName>
    <SystemName> DocumentManager </SystemName>
    <Description> Provide document validation service in document.
</Description>
    <ComputationalAttributes>
        <InherentAttributes>
            <id> magellan.cs.iupui.edu:1610/UserValidationServer</id>
            <Version> 1.0 </Version>
            <Author> Zhisheng Huang</Author>
            <Date> August 2002 </Date>
            <Validity> Yes </Validity>
            <Atomicity> Yes </Atomicity>
            <Registration> magellan.cs.iupui.edu:1310/HeadHunter
</Registration>
            <Model> Java RMI </Model>
        </InherentAttributes>
        <FunctionalAttributes>
            <Purpose> Act as validation server for users in document.
</Purpose>
            <Algorithms>
                <algorithm> JFC </algorithm>
            </Algorithms>
            <Complexity> O(1) </Complexity>
            <SyntacticContract>
                <ProvidedInterfaces>
                    <Interface> IValidationCase1 </Interface>
                </ProvidedInterfaces>
                <RequiredInterfaces>
                </RequiredInterfaces>
            </SyntacticContract>
            <Technologies>
                <technology> Java RMI </technology>
            </Technologies>
            <ExpectedResources>
                <resource> CPU: 500 mhz </resource>
                <resource> Memory: 1.0 Gb </resource>
            </ExpectedResources>
            <DesignPatterns>
                <pattern> </pattern>
            </DesignPatterns>
            <KnownUsage>
                <usage> </usage>
            </KnownUsage>
```

Table 2.1 UMM specification template [ANJ04]

```
            <Aliases>
                <alias> </alias>
            </Aliases>
       </FunctionalAttributes>
    </ComputationalAttributes>
    <CooperationAttributes>
        <PreprocessingCollaborators>
            <Collaborator> DocumentTerminalCase1 </Collaborator>
        </PreprocessingCollaborators>
        <PostprocessingCollaborators>
        </PostprocessingCollaborators>
    </CooperationAttributes>
    <AuxiliaryAttributes>
        <Mobility> No </Mobility>
        <Security> L1 </Security>
        <FaultTolerance> L1 </FaultTolerance>
    </AuxiliaryAttributes>
    <QoS>
        <QoSMetrics>
            <Metric>
                <ParameterName> throughput </ParameterName>
                <FunctionName> validate </FunctionName>
                <Value> 675 </Value>
            </Metric>
            <Metric>
                <ParameterName> endToEndDelay </ParameterName>
                <FunctionName> validate </FunctionName>
                <Value> 1541 </Value>
            </Metric>
        </QoSMetrics>
        <QoSLevel> L1 </QoSLevel>
        <Cost> L1 </Cost>
        <QualityLevel> L1 </QualityLevel>
    </QoS>
</UMM_ConcreteComponent>
```

Table 2.1 UMM specification template (cont’d)

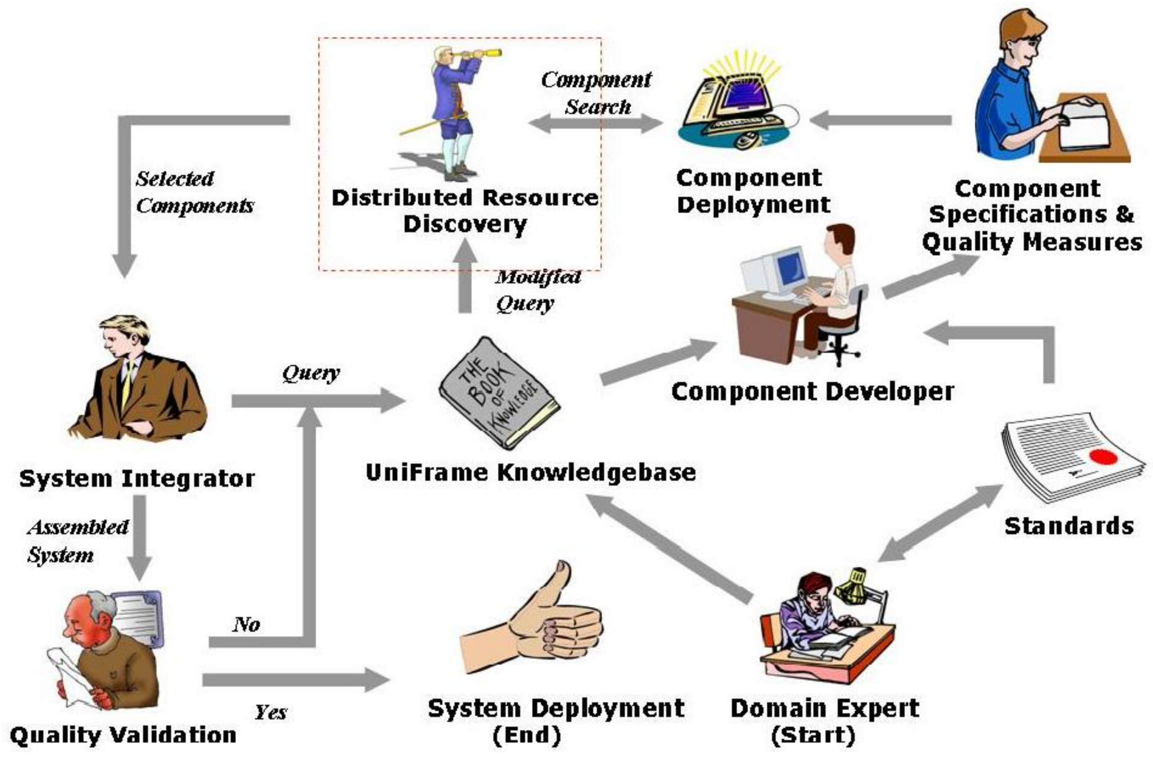


Figure 2.1. The UniFrame Approach

The generation of a system from individually developed and deployed components, starts with a system integrator submitting a query for a specific DCS to the UniFrame Process. Using the domain knowledge, a set of functional and QoS-based search parameters are extracted from the query in the form of UMM descriptions and then submitted to the URDS. The *head-hunters* of the URDS aid in discovering the components which meet the functional and QoS requirements. The discovered components are returned to the system integrator, who then selects the components and asks the UniFrame generator to construct the DCS. Before the system it is deployed, it is tested for its desired functionality and QoS requirements. If the functionality or QoS requirements are not met, then another system is built from the collection of components and is tested in an iterative manner. The URDS, thus, forms a critical piece in the UA by providing the appropriate components to the system integrator.

2.1.3. UniFrame Resource Discovery Service (URDS)

The URDS encompasses of the following entities: a) Internet Component Broker (ICB), b) Headhunters (HHs), c) Meta-Repositories, d) Active Registries, e) Services (C1… Cn), and f) Adapter components (AC1...ACn). Figure 2.2 shows interactions between different components of the URDS architecture.

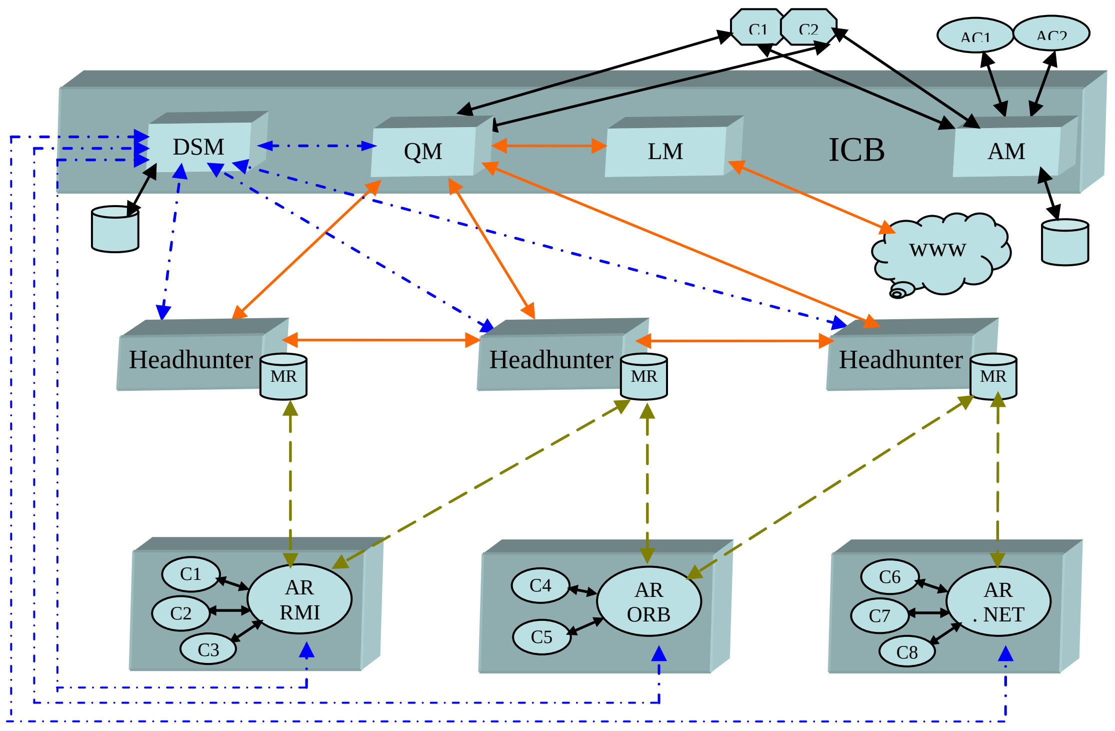


Figure 2.2. URDS Architecture [DEV05]

a) *Internet Component Broker (ICB):* The ICB is a collection of four sub-components the Domain Security Manager (DSM), the Query Manager (QM), the Link Manager (LM) and the Adapter Manager (AM). The ICB constitutes the communication infrastructure necessary to locate and identify services, enforce domain security and to handle the negotiation between heterogeneous components. All the services provided by the ICB are reachable at well-known locations. It is assumed that there are fixed number of ICBs, deployed at well-known locations hosted by corporations or organizations supporting UniFrame.

- *Domain Security Manager (DSM):* The DSM functions as an authorized third-party that provides security and integrity of multicast announcements, which are necessary for the discovery process of URDS. The DSM is responsible for the secret key generation and distribution for ICB. It also enforces group memberships and access controls to multicast resources through authentication and the use of access control lists (ACL).

- *Query Manager (QM):* The QM is responsible for translating a system integrator's/component assembler's requirement into a Structured Query Language (SQL) statement and dispatching the query to the 'appropriate' Headhunters (HHs). These HHs are selected based upon the domain specified in the SQL query and also based on the profile information maintained by the QM [DEV05]. The QM in-turn receives a list of service provider components which match the search criteria. The QM in conjunction with the LM is responsible for propagating the queries to other linked ICBs.

- *Link Manager (LM):* The LM is responsible for establishing links between ICBs to form a federation, and to propagate queries received from QM to other ICBs. The LM is configured by an ICB administrator with the location information of LMs and other ICBs, which helps in establishing links.

- *Adapter Manager (AM):* The AM serves as a registry/lookup service for clients seeking adapter components. These components act as the glue-wrapper [TUM04], allowing interoperability among off-shelf components chosen to implement the distributed system. The adapter components register with the AM and while doing so they indicate their specialization, i.e., which component models they can bridge efficiently. The AM is also capable of dynamic and semi-automatic generation of glue [TUM04], which help in bridging heterogeneous service components. The system integrator/component developer can contact the AM for adapter components which match their needs.

b) *Headhunter (HH):* The HH is one of the principal [SIR03] entities of URDS. It is responsible for discovering service providers and registering the functionality of these service providers. HHs returns a list of service providers to the ICB that matches the requirements of the system integrator's/component assembler's request forwarded by the QM.

c) *Meta-Repository (MR):* The MR serves as the data repository for a HH, it stores the UniFrame specification information (meta-data) of the service components which adhere to different heterogeneous models.

d) *Active Registry (AR):* The AR serves as a registry/lookup service for a particular distributed computing model (such as RMI, .NET, CORBA, etc.), and is extended to listen and respond to multicast messages from HHs.

e) *Services (S1..Sn):* The services may belong to heterogeneous component models such as RMI, .NET, CORBA, etc. Each of these services is identified by its service type name and its UMM Specification.

f) *Adapter Components (AC1..ACn):* Adapter Components serves as bridges between heterogeneous components. These components can by dynamically synthesized by AM using glue-wrapper technology [TUM04].

The URDS architecture is organized into a federated hierarchy of ICBs and HHs, thereby promoting scalability and fault handling [MYS04]. Each ICB consists of one level of zero or more HHs attached to it and all ICBs that are participating in the URDS are unidirectionally linked to form a federated group. The discovery process in URDS is based on an administratively defined application domain (such as Heath Care, Banking or

Manufacturing) and each domain is supported by the sector or organization providing the URDS service. Finally, the URDS discovery process is based on periodic and secure multicast announcements.

In URDS, there is no controlling authority that keeps track of the behavior of the system, because of which the system administrator is not completely aware of the activities in the system. A monitoring system would enable control and coordination of such activities of the URDS. The following section discusses different approaches to monitor distributed systems.

2.2. Monitoring and Management Systems

The monitoring is defined as the process of dynamic collection, interpretation and presentation of information concerning objects or software processes under analysis [JOY87]. The monitoring of a distributed system is critical. It facilitates a variety of tasks such as the representation of information, fault handling, security and system management. The following sub-sections give a brief description about some of the existing monitoring systems, and Chapter 3 discusses in detail different monitoring systems and their behavior in a distributed computing scenario.

2.2.1. Globus Toolkit : Monitoring and Discovery Service

The Monitoring and Discovery System (MDS) [MDS2] is the information services component of the Globus Toolkit [GLO]. It provides details about the available resources on the Grid [FOS99] and their statuses. MDS2 is a version of MDS, which is built upon the Lightweight Directory Access Protocol (LDAP) [LDA], and uses an extensible framework for monitoring static and dynamic information, such as the status of a Grid and its components, networks, and storage systems.

MDS2 is primarily used to deal with the resource selection problem of Grid, i.e., to identify the host or a set of hosts on which user can run the application. MDS2 provides a uniform and easy to use interface, for monitoring of the data collected by the lower-level Information Providers (IPs) [CZA01]. It has a de-centralized structure, which helps in scaling and in handling static and dynamic data about resources and associated queues.

2.2.2. Hawkeye

Hawkeye [HWA] is a monitoring tool developed by the Condor group, utilizing Condor [CON] and ClassAd [CLA] core technologies. It has been designed to monitor various attributes of a collection of systems. It automates the problem detection and software maintenance in a DCS. For example, Hawkeye identifies high CPU load, high network traffic, and resource failure with in a distributed system.

The architecture of Hawkeye comprises four major components: Hawkeye pool, Hawkeye manager, Hawkeye monitoring agent, and Hawkeye module. These components are organized in a four-level hierarchical structure. In a set of computers (known as a “pool”), one computer serves as Hawkeye Manager and the rest function as Hawkeye monitoring agents. The Hawkeye Manager serves as the head of the pool; it collects all information and handles user queries. A Monitoring Agent is a distributed information service component that collects ClassAds (which are a set of attribute value pairs) and integrates them and sends the result to the Manager at specified time intervals. A Hawkeye Module is a simple sensor that announces the resource information in a ClassAd format. The above mentioned architecture aids in easy monitoring and software maintenance within a pool. Hawkeye is designed to gather information such as memory usage, system load, disk space, and other hardware resources; it has not been designed to monitor software resources.

2.2.3. Ganglia

Ganglia [MAS04] is a Java-based, scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters, relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state.

Ganglia consists of three modules, namely *gmond* (Ganglia Monitoring Daemon), *gmetad* (Ganglia Meta Daemon) and Ganglia *web front-end*. The *gmond* provides monitoring on a single cluster by implementing the listen/announce protocol and responding to client requests by returning an XML representation of its monitoring data. *gmond* runs on every node of a cluster. On the other hand, *gmetad* provides a federation of multiple clusters. A *gmetad* service can be configured in such a way that it sends data it received to other *gmetad* services. In that way, it is possible to establish monitoring of several clusters. Ganglia's *web front-end* allows users to see graphs and all historical data saved in Round-Robin databases, allowing all cluster and host metrics to be viewed in real-time. Ganglia is designed to monitor information such as CPU load, memory and other hardware related resources; it does not monitor software resources.

2.2.4. Supermon

Supermon [SOT02] is a distributed monitoring system for clusters that causes insignificant load on the client and provides a flexible set of tools for this purpose.

Supermon uses a client-server data protocol based on symbolic expressions (S-expressions) at all levels of Supermon, i.e., from individual nodes to entire clusters. Supermon mainly consists of three different components:

- A loadable kernel module providing data (CPU information, memory information, etc.), in the form of S-expressions,
- A single node data server (Mon) that serves data prepared by the kernel module,
- A data concentrator (Super-mon) which combines samples from various nodes into a single data sample through a TCP port.

Supermon uses a modified version of the SunRPC *rstat* protocol to collect data from remote cluster nodes. This protocol is based on S-expressions instead of XML with the advantage that it can operate in a heterogeneous environment. Supermon also provides a fast and efficient data collection capability. Supermon is designed to monitor the hardware resources; it lacks the capability of dealing with software resources.

2.2.5. Relational Grid Monitoring Architecture

The Relational Grid Monitoring Architecture (R-GMA) [DAT02] is a Grid Information and Monitoring System which has been developed within the European DataGrid Project [DAT02]. R-GMA is an implementation of Grid Monitoring Architecture (GMA) [TIE02] within the Global Grid Forum (GGF) [GGF], and it is based on the Relational Database Management System (RDBMS) [FIS01] and the Java Servlet technology. It is mainly used where notification of events is required, i.e., a user can subscribe to a flow of data with specific properties directly from a data source. For example, a user can subscribe to a load-data data stream, and create a new Producer/Consumer pairing to allow notification when the load reaches some maximum or minimum.

The GMA is an architecture for monitoring entities that show the characteristics of Grid platforms. The GMA [TIE02] consists of three entities, namely Consumers, Producers and a Registry. In the GMA, the producers register themselves with the registry and describe the type and structure of information they want to make available to the Grid. Consumers can query the registry to find out what type of information is available and locate producers that provide such information. Once this information is known, the consumer can contact the producer directly to obtain the relevant data. By specifying the consumer/producer protocol and the interfaces to the registry, one can build inter-operable services.

The R-GMA architecture closely maps with that of GMA architecture. In R-GMA, when a Producer is created, its registration details are sent to the Registry via the *ProducerServlet*. The Registry records details about the Producer, which include description and view of the data published (table name and the row(s) of a table). The *ProducerServlet* registers this information in the RDBMS, which holds the information of all Producers. The *ConsumerServlet* of the Consumer consults the Registry to locate suitable Producers. The Registry then returns a list of Producers to the *ConsumerServlet* which matches Consumer's interests. Then the Producer and Consumer directly communicate via their respective Servlets.

2.2.6. Monitoring of Software Components

[GAO00] discusses component traceability and maintenance issues and solutions in supporting software components and component-based systems. It provides a Java framework to track and monitor software components in component-based systems. It deals with both in-house and third party components. It also introduces the concept of traceable software components and a new event-based tracking model for the proposed framework. Traceability of software component refers to the extent of its built-in capability of tracking the status of component attribute and behavior. The event-based tracking model is a systematic mechanism that supports engineers in monitoring and checking the behaviors of components and their interactions in a component-based program. [GAO00] addresses different classifications for component traces, such as operational traces, performance traces, state traces, event traces and error traces. It also discusses component traceability and explains how to construct components which can be monitored, to check their status, performance, behavior and interactions among them. URDSMMS employs the performance trace and error trace approaches to show the performance of the system and errors for a particular entity of the URDS.

[DIA00] presents a framework for monitoring component interactions. This framework supports the development and testing of distributed software applications. The monitoring framework is part of a larger framework which is built on top of a CORBA distributed processing environment, which supports development and testing of distributed software applications. This framework has been designed and implemented in close co-operation between the FRIENDS (FRamework Integrated ENgineering and Deployment of Services) project [FRI] and the AMIDST (Application of Middleware for Services in Telimatics) [AMI].

The proposed framework in [DIA00], considers an “Object Management Group (OMG) Interface Definition Language (IDL) specification as a contract for distributed interactions and allow a precise monitoring of interaction activities between the application components”. It addresses the monitoring by observing the system behavior at specific observation (or monitoring) points and effectively representing the information in a graphical or textual manner. The above prototypical monitoring framework is used to test services developed for large application frameworks [FRI].

The systems mentioned in subsections 2.2.1 – 2.2.5, dealt with the monitoring of hardware related resources. These frameworks were implemented to support large scale clusters and grids and, hence, do not focus on software components discovery. The frameworks in subsection 2.2.6 deal with monitoring software components ([GAO00]), particularly in component based systems [DIA00] deals with component interactions. Though these prototypes were implemented for distributed systems, there is no work pertaining to component discovery. The URDSMMS is a prototype for monitoring the URDS, which uses some of the monitoring principles of the above related work (this will be later discussed in chapter 3).

This chapter presented a brief overview about the UniFrame and URDS. The chapter also discusses some of the related monitoring systems and the approaches for monitoring a DCS. The next chapter provides an overview of monitoring in distributed systems, the challenges faced in it and the various features of URDSMMS.

3. MONITORING DISTRIBUTED SYSTEMS

The previous chapter gave an overview of UniFrame and relevant related work associated with URDSMMS. This chapter provides details about monitoring systems, its importance and various challenges involved with a monitoring system in a distributed environment and different approaches followed to monitor distributed systems. The chapter also discusses about the features that have been implemented in URDSMMS and the importance of these features.

3.1 Importance of Monitoring

Monitoring can be defined as the process of dynamic collection, interpretation and presentation of information concerning objects or software processes under scrutiny [JOY87]. It is the process of collection of dynamic behavioral data during program execution, interpreting and analyzing that data and displaying the processed information to the user in form of text, graphs or animation.

Monitoring is an essential and efficient way of obtaining information in a distributed component system, where managerial decisions can be made and also the behavior of it can be observed. Monitoring information can also help in observing the component interactions. In a DCS, monitoring helps in a wide variety of tasks, such as testing, debugging, visualization system management, security, fault management, performance management and configuration [SCH95, MAN95].

3.2 Challenges of Monitoring in Distributed Systems

The distribution of entities in a DCS makes its monitoring more complicated than a centralized system. This is due to the fact that a typical distributed system may consist of a number of cooperating processes, executing on different remote machines and communicating via message passing. Moreover monitoring requirements are not static in either a distributed or a centralized system; but evolve as the system may be dynamic. Thus, there may be a need to dynamically change the patterns of activities which are monitored without having to bring the operation of the underlying system to a halt.

Additionally, there are a number of reasons why the debugging, testing and evaluation of distributed systems is more difficult when compared to the same activity in sequential programs:

- A distributed system has many foci of control. Thus the sequential monitoring and debugging techniques, such as tracing and breakpoints, based on process state and program counter, need to be extended to be applicable to distributed systems [JOY87].
- Delays in transferring of information mean this information may be out of date.
- Communication delays among nodes in a distributed system, makes it difficult to determine a global view of the system status.
- Due to large number of objects which generating monitoring information, collecting all the information to monitor becomes a difficult task.
- It is possible that the distributed system may be heterogeneous in nature and a canonical form is needed to encode messages passed between these heterogeneous systems [SCH95].

URDSMMS addresses a subset of the above mentioned problems. URDSMMS is built over a discovery service (URDS), which is a distributed system by itself. The monitoring system collects information from all the entities of this DCS based on the process state and message communication between entities. URDSMMS is designed to collect information from a large number of objects. The monitoring system does not consider any, message losses, out of date or out of order events [LAM78], or heterogeneity at this point (further discussed in chapter 6).

3.3 Classification of Monitoring Systems

There are three broad approaches for the classification of monitoring systems, namely, hardware, software and hybrid monitoring systems.

In hardware monitoring, a specialized hardware is essential to monitor the system and detect the events. The hardware monitoring performs event detection by having sensors or a detection circuitry. In such a case, it is difficult to relate recorded events in the monitored program, i.e., to find the problem-oriented reference [HOF94]. Hardware monitoring systems have the advantage of being non-intrusive, and being efficiently accurate, but such monitoring systems require external hardware resources, which limits the amount of interference with the monitoring system.

The software monitoring systems uses software programs to observe the monitored entities. This requires system resources such as the CPU and memory. These monitored programs are usually instrumented by inserting monitoring instructions called software probes to collect information. In this approach, each monitored event can be associated to a point in a program. Software systems are more flexible and portable as they provide an easy construction and easy portability to other platforms. However, software monitoring is intrusive in nature, as it has to share the system resources with the monitored system.

The hybrid monitoring systems attempts to combine the advantages of hardware and software systems. It brings together the non-intrusive nature of hardware approaches and the flexibility of the software approaches. ZM4 [HOF94] is an example of hybrid monitoring system that allows monitoring programs to be evaluated for performance and program behavior to be observed.

The present monitoring system (URDSMMS) uses the software monitoring approach to provide flexibility and platform portability.

3.4 Monitoring Types

The basic types of monitoring approaches based on the information collection are the *Time-driven monitoring* (sampling) [HOF94] and the *Event-driven monitoring* (tracing) [HOF94].

Using sampling, information about the monitored system is collected synchronously (at a specific time rate), or asynchronously (through direct request of the monitoring system). On the other hand, in event-driven monitoring, information is collected when an event occurs in the system.

The event-driven monitoring consists of the reporting of all occurrences of an event within a certain interval of time. It is synchronous with the occurrence of an event; it is performed when all occurrences of an event must be known (e.g., when collecting history information) or when each occurrence of an event must be followed by a certain action. On the other hand, sampling is the collection of information at the request of the monitor. Sampling may be asynchronous with the occurrence of an event.

The event-driven monitoring is more intrusive then time-driven because tracing is synchronous with the occurrence of an event. The time-driven monitoring provides statistical information about the program execution, whereas event-driven monitoring represents the behavior of a program by a sequence of events. Some monitoring systems implement both monitoring types to satisfy the varying requirements and constraints [MAN95]. The URDSMMS uses both time-driven monitoring and event-driven approaches so as to collect monitoring data asynchronously. It uses event-driven approach to handle the events and uses a time-driven approach for updating the web interface.

3.5 Monitoring in URDS

The existing URDS architecture provides a framework for the discovery of software components for constructing a DCS. Currently, URDS is not provided with a monitoring system, which would help in a better management and coordination of its entities. The following section discusses different monitoring scenarios for the URDS. The features that should be incorporated in the URDS for its monitoring and the benefits obtained in considering these features are discussed.

3.5.1 Reasons for Monitoring URDS

The URDS consists of many entities which collectively account for the resource discovery process in UniFrame. Among these entities, numerous messages are passed concurrently for the discovery of software components. To control and coordinate such concurrent activities in URDS, monitoring is essential.

Monitoring assists the system administrator with the knowledge about the activities on the local and remote machines. It provides consolidated, abstracted and dynamic data; this is represented in the form of text or dynamic graphs. Monitoring can determine the performance of the overall system, such as the average response time for discovering the software components. Monitoring also helps the administrator in knowing the number of active entities (Headhunters, Active Registries) in the system.

Monitoring of the URDS can also include some system management activities such as, initiation and termination of resources (entities) according to the necessity, which may help in improving the response time for discovering components. Using this monitoring interface, the system administrator should be in a position to take corrective steps, in case of errors or failures. Monitoring via an easy-to-use interface improves the usability of URDS and helps in easy understanding of the system for a new user. The following section describes the monitoring features of URDSMMS.

3.6 Features of the URDS Monitoring and Management System (URDSMMS)

The URDSMMS is provided with two views; administrator view and system integrator view. These two views help in providing a basic separation of concerns and address the issues of access privileges to the URDS. The URDSMMS provides supervisory features for the administrator, allowing to monitor and manage all the activities of URDS, whereas the system integrator is provided with only restricted features, allowing him to submit a query and retrieve results. Necessary secure login is provided to distinguish between the administrator and the system integrator.

3.6.1 System Administrator View

This subsection provides details about the various features that are considered in the *System Administrator View* and how they are helpful for controlling and coordinating activities in URDS.

Start Entities: The system administrator is given an option of running different entities of URDS using a GUI. This feature helps the administrator in starting the Domain Security Manager, Query Manager, Headhunters and Active Registries at any remote/local location. This avoids the administrator having to run the entities using a console, which is cryptic and error-prone.

URDS Snapshot: This gives the statistical details of all entities currently active in URDS. Details such as the location of entities, number of Headhunters or Active Registries currently active and number of queries currently being processed by the system can be known. An option is provided for the dynamic display of attributes (such as the name, the location and the domain) of certain entities (e.g., Headhunters and Active Registries). This feature also displays the ranking [DEV05] associated with each Headhunter, which might help the Query Manager is selecting the best Headhunter and propagating the query to it. This ranking is given by the Query Manager based on the components returned by Headhunter for each query.

Message Traffic: In URDS, different kinds of message communications are possible among its constituents (as described in Chapter 2). Because of this traffic, a system administrator may be interested to know a classification of the traffic. The messages are classified into three broad categories, namely *authentication, query processing* and *component update.* The authentication messages consist of the principals (Headhunters and Active Registries) and Query Manager (QM) contacting the Domain Security Manager (DSM) for authentication. The query processing consists of the System Integrator (SI) issuing the query to QM, which in-turn passes it to Headhunters and retrieves the components list. The software components (or services) registering to the Active Registry and the Active Registry propagating them to the Headhunters, fall under the third category, the component update. The above classification helps the system administrator in knowing what activity is dominant at a particular time in the system and thereby controlling the traffic causing parameters as necessary.

Average Response Time Graph: The administrator view is provided with a graph which represents the average query response time of the URDS. The graph depicts the response time for the most recent 20 queries. This graph also helps in determining the performance of the system. If the response time is too high, then maybe the load on the system is too high. Thus, such a graph can assist in identifying bottlenecks in the URDS, which could be addressed by the system administrator.

Utilization of Query Manager: This feature is included so as to know the percentage of time the QM is actually busy. If the QM’s utilization is high, the administrator should be able to control the number of new queries being issued (by blocking or by sending a message to the system integrator). The system administrator can also take a decision of initiating new Headhunters and or QMs, if necessary.

Utilization of Headhunter: This feature is is an indication of the busy nature of a particular Headhunter. If a Headhunter is idle for a long time, then there is a greater possibility that the Headhunter is under performing, or no query is being propagated to that particular Headhunter. In such as case, the system administrator can take a decision of terminating that particular Headhunter. On the other side, if a Headhunter's utilization is high for a long period of time, the system administrator can take other corrective action, such as initiating another Headhunter on the same remote/local machine.

Stop Entities: The system administrator is provided with an option of terminating entities in the URDS. This decision could be made if few hardware resources are available. This feature is mainly highlighted when a particular Headhunter's utilization is low for a prolonged period of time or that Headhunter's services are no longer required. The same scenario can be applied to a Active Registry also. The terminating of the Domain Security Manager and Query Manager comes in to the picture if multiple URDS are deployed and this monitoring system is used to control all of them together.

Error Monitor: The feature is helpful in catching exceptions and displaying them to the system administrator. These messages assist the administrator in taking some corrective decision if necessary. Additionally, error help is also provided which assists the system administrator.

3.6.2 System Integrator View

The *System Integrator View* provides a GUI for the user to submit queries and view the results. The following features are provided in this view and their importance is indicated.

Submit Query: This feature allows the system integrator to submit queries for the search of components. While submitting the queries for processing, the system integrator is given an option of entering the response time, also, within which the results should be retrieved. Each query is associated with a unique query id, with which the system integrator can retrieve results.

View Results: Once the query has been processed, the results can be viewed by submitting the query id. The result for each query consists of the number of components matched, the time taken for processing the query (response time), a list of component names and the ids associated to each component. Along with these details, a feedback [DEV05] for each component is also displayed, which helps in selecting best matched components.

Help: A navigational help is provided in the URDSMMS (for both the views), which aids a user/administrator in better navigation, and it also in understanding how to use the different features of the system. Help for errors are also provided in the system administrator view.

This chapter provided an overview of monitoring systems and associated challenges in creating them for a distributed computing environment. The chapter described different features that are considered in the design of the URDSMMS. The next chapter discusses about the design and implementation details of the URDSMMS.

4. DESIGN AND IMPLEMENTATION OF URDSMMS

The chapter 3 provided an explanation of monitoring systems in a distributed computing system and an explanation of different features for monitoring in URDSMMS. This chapter discusses the URDSMMS architecture in detail, focusing on its high-level design, the algorithms and the implementation details of the mentioned features of URDSMMS.

Figure 4.1 illustrates the constituents of the URDSMMS and their interactions with the different entities in URDS.

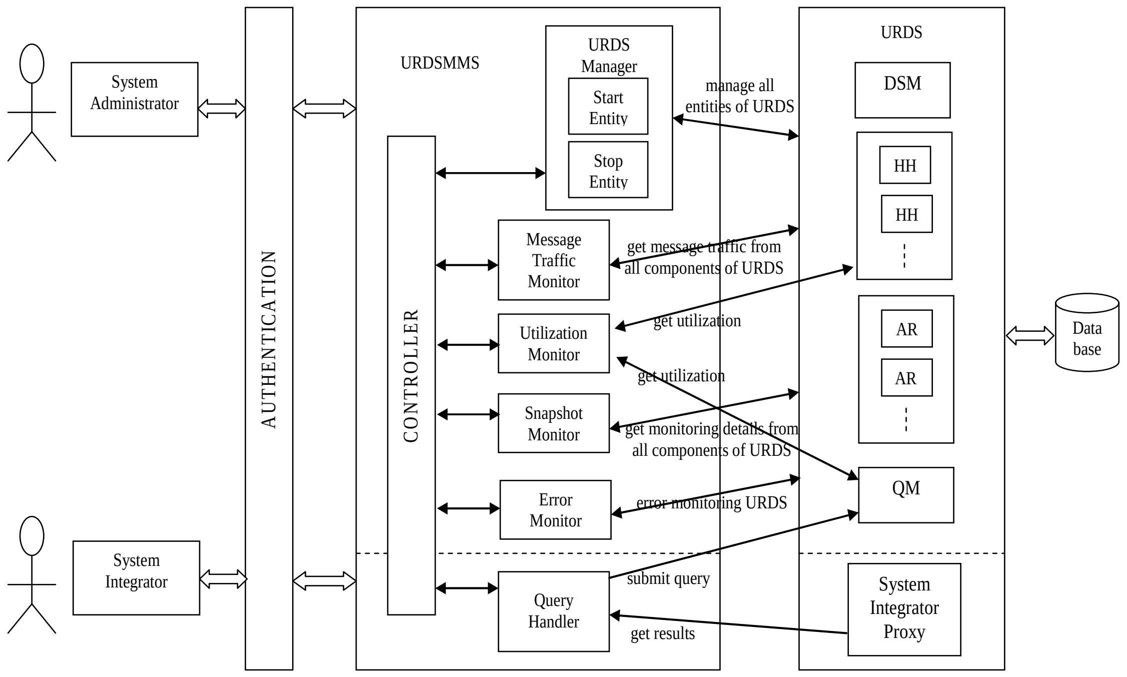


Figure 4.1: URDS Monitoring and Management System Architecture

The above architecture is shown with two types of users for URDSMMS, thus leading to two views (described in the previous chapter), called the system administrator view and the system integrator. URDSMMS architecture allows a comprehensive set of features to the administrator, thereby allowing him to monitor and manage all the activities of the URDS. The system integrator is provided with a few restricted features (because of the security reasons), allowing him to submit a query and retrieve the results. A necessary secure login is provided to distinguish between the system administrator and the system integrator. In order to determine the specific constituents under each view, the URDS has to be looked in terms of state diagrams.

The state diagrams [ARL2002, OES99] of the constituent elements of URDS depict the various state changes of each element of the URDS. The event, triggers the transition between these states (state changes). This analysis provides an insight into the interval and the set of events to be recorded by the URDSMMS. Appendix A shows the state diagrams for each entity of the URDS system.

4.1 Constituents of the URDSMMS

The features of monitoring systems which were mentioned in chapter 3 can be grouped into different categories, namely: i) Controller ii) URDS Manager, iii) Message Traffic Monitor, iv) Utilization Monitor, v) Snapshot Monitor, vi) Error Monitor, and vii) Query Handler. The grouping is based on the functionality of each constituent and on the separation of concerns. The following subsections discuss the design details of each of these constituents and their interactions with URDS and other constituents of the URDSMMS.

4.1.1 Controller

The *Controller* acts as a gateway between users (System Administrator/System Integrator) and the URDSMMS. Depending upon the requested service by the users, the controller redirects the request to the appropriate component of URDSMMS.

Algorithm for CONTROLLER initialization: In this algorithm the *CONTROLLER* initializes all the services, i.e., the *URDS MANAGER, MESSAGE TRAFFIC MONITOR, UTILIZATION MONITOR, SNAPSHOT MONITOR, ERROR MONITOR* and *QUERY PROCESSOR* by calling their initialization methods.

```
CONTROLLER_INITIALIZATION
BEGIN
        CALL URDS_MANAGER_INITIALIZATION
        CALL MSG_TRAFFIC_MONITOR_INITIALIZATION
        CALL UTIL_MONITOR_INITILIZATION
        CALL SNAPSHOT_MONITOR_INITILIZATION
        CALL ERROR_MONITOR_INITILIZATION
        CALL QUERY_HANDLER_INITILIZATION
END
```

4.1.2 URDS Manager

The *URDS Manager* is responsible for all the management activities such as initiating and terminating the entities of URDS (DSM, HH, AR, and QM). Depending upon a successful/unsuccessful creation or termination an appropriate message is displayed with the help of the *Controller*.

Algorithm for URDS Manager initialization: This algorithm outlines the process of *URDS Manager* activating the *START_ENTITY_SERVICE* and *STOP_ENTITY_SERVICE*.

```
URDS_MANAGER_INITIALIZATION
BEGIN
        ACTIVATE START_ENTITY_SERVICE
        ACTIVATE STOP_ENTITY_SERVICE
END
```

Algorithm for initiating an entity: This algorithm outlines the process of creating new instances of URDS entities. Depending upon the system administrator’s request, the appropriate service is invoked. In the below algorithm, *entityType* refers to the type of entity (DSM, HH, AR or QM). *status* is a boolean flag for displaying appropriate (successful/unsuccessful) message on the interface, *Attributes* is a set of parameters and these parameters depend upon the type of entity that is being initiated.

```
START_ENTITY_SERVICE
BEGIN
    IN: entityType, Attributes
    IF entityType is DSM
        status = CALL DSM_START_SERVICE (entityType, Attributes)
    END IF
    ELSE IF entityType is HH
        status = CALL HH_START_ SERVICE (entityType, Attributes)
    END ELSE IF
    ELSE IF entityType is AR
        status = CALL AR_START_ SERVICE (entityType, Attributes)
    END ELSE IF
    ELSE IF entityType is QM
        status = CALL QM_START_ SERVICE (entityType, Attributes)
    END ELSE IF
    ELSE
        // set flag to false value
        SET status to FALSE
    END ELSE
    // to display successful/unsuccessful message on the interface
    DISPLAY status
END
```

For all the routines invoked in *START_ENTITY_SERVICE*, a generic algorithm is presented with different set of parameters for each type of entity. In the below algorithm *status_flag,* is a boolean flag which denotes a successful initiation or a failure. *Location* is passed as *Attributes* for the DSM. The *Attributes* for HH include *Location*, *dsmLocation,* *hhName, hhPwd* and *Domain*. *Location, dsmLocation, arName, arPwd* and, *Domain* comprise the *Attributes* for the AR; and *Location, dsmLocation, qmName* and *qmPwd* denote the *Attributes* for the QM. *DEPLOY,* this command deploys that particular entity (DSM, HH, AR or QM) on the remote host.

```
GENERIC_ALGORITHM_START_ENTITIES
BEGIN
    IN: entityType, Attributes
    OUT: status_flag (success/failure)
    // instantiate entity on the remote host which returns a true or a false flag
    status_flag = DEPLOY entityType on remote host
    IF status_flag is TRUE
        // stores the location where entity is hosted in database
        INSERT entityLocation
        // starts an thread if entity is a HH or a QM
        IF entityType is HH or QM
            START UTIL_SERVICE
        END IF
        // starts an error monitor thread for that particular entity
        START ERROR_MONITOR_SERVICE
    END IF
    // to return flag value to START_ENTITY_SERVICE
    return status_flag
END
```

Algorithm for terminating an entity: This algorithm outlines the process for terminating an existing entity of the URDS. The system administrator contacts the URDSMMS for terminating a particular entity. It then lookups the entity and issues a TERMINATE command, this returns a true flag upon a successful termination else a false value. If the flag value is true it removes the entry from the database and if the *entityType* is HH, AR or QM, the DSM is notified of the appropriate change. In the below algorithm, *entityLocation* represents the location where the entity is running.

```
STOP_ENTITY_SERVICE
BEGIN
    IN: entityType (DSM, HH, AR, or QM), entityLocation
    OUT: status_flag (success/failure)
    IF entityType is DSM
        // obtain handle to DSM
        dsmLocation = LOOKUP entityLocation
        // terminate DSM which returns a true or false flag
        status_flag = TERMINATE DSM
        IF status_flag is TRUE
            // removes the entry from the database
            DELETE dsmLocation
        END IF
    END IF
    ELSE IF entityType is HH
        // obtain handle to HH
        hhLocation = LOOKUP enitityLocation
        // terminate HH which returns a true or false flag
        status_flag = TERMINATE HH
        IF status_flag is TRUE
            // remove the corresponding hhLocation from the DSM’s HHs List
            NOTIFY DSM
            // removes the entry from the database
            DELETE hhLocation
        END IF
    END ELSE IF
    ELSE IF entityType is AR
        // obtain handle to AR
        arLocation = LOOKUP enitityLocation
        // terminate AR which returns a true or false flag
        status_flag = TERMINATE AR
        IF status_flag is TRUE
            // remove the corresponding arLocation from the DSM's AR's List
            NOTIFY DSM
            // removes the entry from the database
            DELETE arLocation
        END IF
    END ELSE IF
    ELSE IF entityType is QM
        // obtain handle to QM
        qmLocation = LOOKUP enitityLocation
        // terminate QM which returns a true or false flag
        status_flag = TERMINATE QM
        IF status_flag is TRUE
            // remove the corresponding qmLocation from the DSM's entry
            NOTIFY DSM
            // removes the entry from the database
            DELETE qmLocation
        END IF
    END ELSE IF
    ELSE
        // set flag to false value
        SET status_flag to FALSE
    END ELSE
    // to display on the on the interface about a successful/unsuccessful termination
    return status_flag
END
```

4.1.3 Message Traffic Monitor

The *Message Traffic Monitor* keeps track of the message traffic information in the URDS. The traffic monitor uses an event-driven approach to update its traffic information this information is updated on the web interface upon the request. This information is classified into three types, namely, authentication messages, query processing messages and components update messages. These messages are recorded by *software probes* [MAN95] when an inter-entity message transfer occurs in the URDS. The software probes are small pieces of code residing within the programs (entities of the URDS) which raise events. These recorded events are responsible for sending messages to the *Message Traffic Monitor*. The traffic monitor in-turn updates its message traffic value(s) depending upon the type of messages (based on the type of event). When the system administrator requests for traffic information the traffic monitor displays information in three categories (authentication, query processing and components update messages).

Algorithms for Handling the Message Traffic: These algorithms outline the process of maintaining the message traffic information. Whenever there is new message, it updates the appropriate counter.

```
MSG_TRAFFIC_MONITOR_INITIALIZATION
BEGIN
        ACTIVATE MSG_TRAFFIC_SERVICE
END

MSG_TRAFFIC_SERVICE
BEGIN
        WHILE TRUE
                /* listens to all URDS (DSM, HH, AR & QM) entities for any message
                communication, this is achieved by using software probes */
                COLLECT trafficInfo
                IF newMessage
                    /* update the authentication / query / components update counter
                    depending upon the type of message */
                    UPDATE authentication/queryPropagation/Component counter
                END IF
                /* system administrator requests URDSMMS for message traffic
                information */
                IF contacted by SystemAdministrator
                    // retrieves the traffic information
                    GET trafficInfo
                    /* display the traffic information on system administrator’s
                    interface */
                    DISPLAY trafficInfo
                END IF
        END WHILE
END
```

4.1.4 Utilization Monitor

The *Utilization Monitor* helps in providing the utilization of the active Headhunters or the Query Manager. As soon as a HH or QM is started, the *Utilization Monitor* starts recording its net busy time (time during which a HH or QM is executing any of its methods). It records the busy time at a regular time intervals, so that a history of values could be provided to the system administrator upon a request.

Algorithm for Initialization of Utilization Monitor: In this algorithm, the utilization monitor activates the *HH_UTIL_SERVICE* and *QM_UTIL_SERVICE*. These algorithms record the utilization of each HH or QM at a regular time interval of 60 seconds (randomly selected).

```
UTIL_MONITOR_INITIALIZATION
BEGIN
    ACTIVATE HH_UTIL _SERVICE
    ACTIVATE QM_UTIL _SERVICE
END
```

Algorithm for Handling Headhunter's Utilization: This algorithm outlines the process of calculating the utilization percentage for each Headhunter and also generates a utilization graph. *netBusyTime* is the net time for the HH to process a query or update its metarepository. The utilization percentage is the net busy time divided by the total time interval for recording the busy time (in this case 60 seconds); precisely *netUtilizationPercentage* = (*netBusyTime*/60)*100 (same formula is used for calculating utilization of QM).

```
HH_UTIL_ SERVICE
BEGIN
    IN: hhLocation
    WHILE TRUE
        // obtain a handle to the HH to record its utilization
        LOOKUP hhLocation
        // obtains the HH's net utilization time, in a particular time interval
        GET netBusyTime
        // calculates the utilization percentage, using the above mentioned formula
        COMPUTE netUtilizationPercentage = (netBusyTime/60)*100
        /* stores the computed utilization values in an ArrayList which
        corresponds to this particular HH */
        STORE value in HHUtilList
        /* system administrator requests URDSMMS for a particular HH's
        utilization information */
        IF contacted by user
            // retrieves the latest set of utilization percentage values
```

```
                GET *HHUtilList*
                /* generate a dynamic graph which shows the utilization
                percentage of HH over a period of time */
                DRAW *HHUtilGraph*
                // displays the graph on the system administrator's interface
                DISPLAY *HHUtilGraph*
            END IF
        END WHILE
END
```

Algorithm for Handling Query Manager's Utilization: This algorithm outlines the process of calculating the utilization percentage for QM and also generates a utilization graph. *netBusyTime* is the time for a QM to check its queue (query input queue), propagate queries to HHs and receive results from HHs.

```
QM_UTIL_ SERVICE
BEGIN
        IN: qmLocation
        WHILE TRUE
            // obtain a handle to the QM to record its utilization
            LOOKUP qmLocation
            // obtains the QM's net utilization time, in a particular time interval
            GET netBusyTime
            // calculates the utilization percentage
            COMPUTE netUtilizationPercentage = (netBusyTime/60)*100
            // stores the computed utilization values in an ArrayList
            STORE value in QMUtilList
            /* system administrator requests URDSMMS for a particular QM's
            utilization information */
            IF contacted by user
                // retrieves the latest set of utilization percentage values
                GET *QMUtilList*
                /* generate a dynamic graph which shows the utilization
                percentage of QM over a period of time */
                DRAW *QMUtilGraph*
                // displays the graph on the system administrator's interface
                DISPLAY *QMUtilGraph*
            END IF
        END WHILE
END
```

4.1.5 Snapshot Monitor

The *Snapshot Monitor* keeps track on the number of active entities in the URDS and the number of queries the URDS is processing. Whenever a new entity is started, the *Snapshot Monitor* updates its values. The *Snapshot Monitor* retrieves the information from the database using a JDBC connection upon the administrator's request. It then propagates the information to the *Controller*, which in-turn updates the values on the interface of system administrator.

Algorithms for Handling the Snapshot of URDS: These algorithms outline the process of obtaining the snapshot of the URDS. Depending upon system administrator's request, details about the HHs and ARs can also be provided, along with the URDS statistics. *enityType* refers to a HH or a AR

```
SNAPSHOT_MONITOR_INITIALIZATION
BEGIN
    ACTIVATE SNAPSHOT_SERVICE
END

SNAPSHOT_SERVICE
BEGIN
```

```
        /* system administrator requests URDSMMS for a snapshot of URDS */
        IF contacted by *user*
                /* retrieves from database details of HH’s, AR’s and the locations of DSM
                and QM */
                GET details from database
                // displays details from database
                DISPLAY *snapshot* of URDS
                // system administrator requests display of specific details of HHs or ARs
                IF requested by *user* for *entityType* details
                        /* retrieves the entity (HH or AR) details from database, such as
                        location, name, domain and rating */
                        GET *entityType_details* from database
                        // display entity details
                        DISPLAY *entityType_details*
                END IF
        END IF
END
```

4.1.6 Error Monitor

The *Error Monitor* provides the error handling part for the URDSMMS. It is responsible for catching the remote exceptions which can occur at any entity of URDS. Exceptions, such as failure of authentication, Headhunter not found, unable to connect, etc., are displayed. When the *Error Monitor* receives an error message from URDS, it immediately dispatches the message to *Controller* which displays the message on the interface of the system administrator. Additionally, an error help is provided corresponding to each error message. The Error Monitor also uses an event-driven approach by using software probes (as described in section 4.1.3), and its values are updated on the interface at regular time intervals.

Algorithms for Handling the Errors of URDS: These algorithms outline the process of catching the exceptions thrown by URDS entities and notifying the system administrator with a relevant error message.

```
ERROR_MONITOR_INITIALIZATION
    ACTIVATE ERROR_MONITOR_SERVCE
END

ERROR_MONITOR_SERVICE
BEGIN
    IN: entityLocation (DSM’s, HH’s, AR’s or QM’s location)
    WHILE TRUE
        /* listens to all URDS (DSM, HH, AR & QM) entities for any remote
        exceptions by using software probes */
        COLLECT errors
        IF errors
            /* displays an error message to the system administrator */
            DISPLAY errorMessage
        END IF
    END WHILE
END
```

4.1.7 Query Handler

The *Query Handler* is responsible for propagating the queries submitted by the system integrator. If a new query is submitted, Query Handler delivers it to QM for processing. The results are retrieved and displayed to the system integrator.

Algorithms for Initialization Query Handler: In this algorithm query handler activates the *NEWQUERY_HANDLER* and *RESULTS_HANDLER.*

```
QUERY_HANDLER_INITIALIZATION
BEGIN
    ACTIVATE NEWQUERY_HANDLER
    ACTIVATE RESULTS_HANDLER
END
```

Algorithms for Handling Queries: These algorithms outline the process of submitting a new query for processing to QM and retrieving results using the RESULTS_HANDLER algorithm and displaying them to the system integrator. The *queryID* is a unique id generated by the monitoring and management system.

```
NEWQUERY_HANDLER
BEGIN
    IN: componentName, responseTime
    // system integrator searches for matching components
    LOOKUP URDS_Proxy
    // obtain a handle to URDS_Proxy to submit a query
    SUBMIT query
END
```

```
RESULTS_HANDLER
BEGIN
    IN: queryID
    // system integrator retrieves results for a particular by submitting the queryID
    GET results from database
    DISPLAY results
END
```

The implementation details for the above mentioned algorithms are provided in subsection 4.4.

4.2 User Interface Design

A good interface is necessary for producing successful software products; it helps in providing better usability for the systems. This section presents the basis for the importance of usability and how usability issues have been tackled in URDSMMS.

"Interface design became important because pleasant, attractive, easy-to-use software sells well." [SUT95]. Usability helps in making the system easy-to-learn and easy-to-use. It describes the quality of a user's (in the present case, either an administrator's or a system integrator's) experience when interacting with a system.

The usability and features of the system were amended by going through a series of steps. Initially the requirements of the target users were identified and these requirements were reviewed and enhanced by having multiple meetings with the UniFrame team members. Also a survey was prepared on the basis of [PRE02]; and was distributed among the UniFrame team to collect their ideas about the interface design and the feature that should be incorporated. Based on the feedback obtained from the survey and the one-to-one meetings, the URDSMMS features such as utilization of HH or QM and average response time were revised and incorporated. The actual copy of the survey and its results are provided in Appendix D.

4.2.1 Use Case Diagrams

The *Use Case* diagrams model the functionality of the system using actors and use cases. [ARL2002, OES99]. A use case diagram displays the relationship among actor(s) and use cases. The roles played by users that use the system are called *actors*. *Use cases* are the actions that an actor can do with the system.

In this subsection, a few use-case diagrams are presented, along with the pre-condition and post-condition of the system, and a brief description about each use case. For

URDSMMS, only two actors (system administrator and system integrator) are considered. Appendix B provides the rest of the use-case diagrams.

Figure 4.2 illustrates a successful/unsuccessful login by a user to access the URDSMMS. The use case starts when the system administrator/system integrator enters the URL. If the correct username and password is entered, the system authenticates the user and redirects to the appropriate interface; else the system displayed an error message and ask to re-enter the username and password.

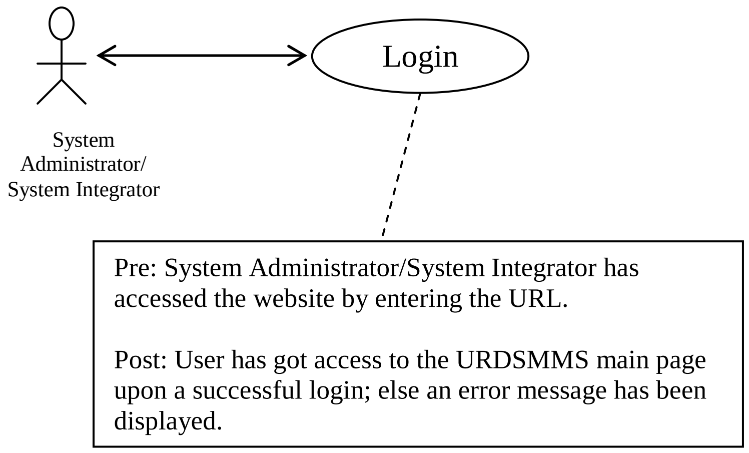
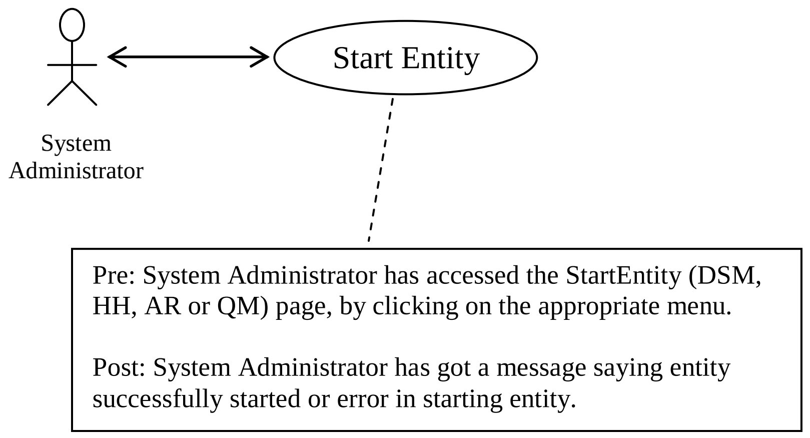

Login
System Administrator/ System Integrator
Pre: System Administrator/System Integrator has accessed the website by entering the URL.
Post: User has got access to the URDSMMS main page upon a successful login; else an error message has been displayed.

Figure 4.2 Use Case Diagram for a successful/unsuccessful login

Figure 4.3 illustrates a use case diagram for starting an entity of URDS. The system administrator is asked to enter the details (depending upon the entity, information such as host name, port number, DSM location, domain, username and password). After checking the appropriateness of the details, a new entity is created and a corresponding status message is displayed. Depending upon the appropriateness, the message can be either a “successful creation” or “error in creation”.



Pre: System Administrator has accessed the StartEntity (DSM, HH, AR or QM) page, by clicking on the appropriate menu.

Post: System Administrator has got a message saying entity successfully started or error in starting entity.

Figure 4.3 Use Case Diagram for starting an entity

4.3. Implementation of the URDSMMS

The following sections describe a prototypical implementation of the URDSMMS architecture using Java and Java-based technologies.

4.3.1 Technology

Before the description of the actual implementation, the technologies that are used for the prototype are briefly described.

4.3.1.1 Java^{TM} 2 Platform Enterprise Edition (J2EE^{TM})

The prototype implementation is based on the architectural model laid out by [SUN01] in J2EE^{TM}. J2EE^{TM} defines a standard that applies to all aspects of architecting, developing and deploying multi-tier server-based applications. The Figure 4.4 [SUN01a] shows the components of the J2EE^{TM} Model.

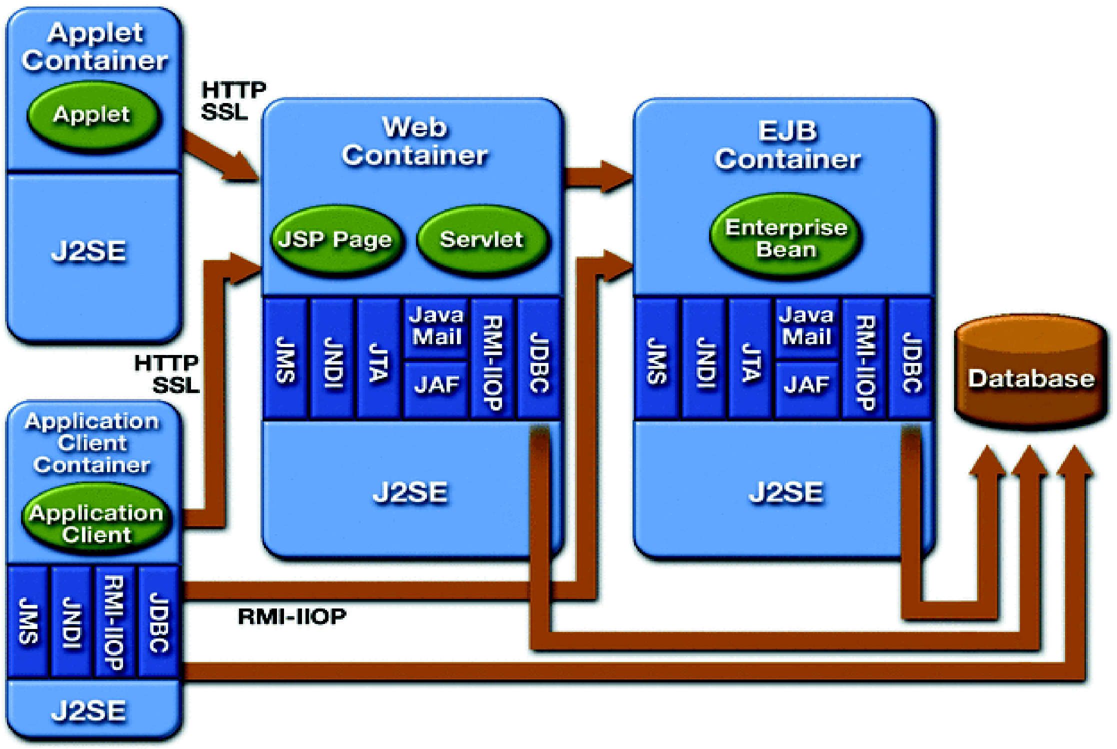


Figure 4.4 Components and Containers of J2EE Model. [SUN01a]

The J2EE™ platform specifies technologies to support multi-tier enterprise applications. [SUN01] categorizes these technologies into three, namely, Component, Services and Communication. The following subsections outline the technologies (in Figure 5.1) in each of these categories that have been used for implementation. (The Java-based technologies described in these sections are proprietary technologies of SUN Microsystems [SUN01])

4.3.1.1.1 Component Technologies

The J2EE™ Component technologies have been used in the prototype to create the front-end client components and back-end service components. The prototype supports two types of clients: *Application Clients* which are structured as *Application Client Components* (described in section 4.3.1.1.1.1) and *Web Clients*, which interact with the *Web Components* (described in section 4.3.1.1.1.2). These two types of J2EE™ components used in the prototype are discussed below:

4.3.1.1.1.1 Application Client Components

Application Clients are client components that execute in their own Java virtual machine. Application clients are hosted in an *Application Client Container* (Figure 5.1).

4.3.1.1.1.2 Web Components

A Web Component is a software entity that provides the necessary response to a request. The J2EE TM platform contains two types of Web Components, namely, *Servlets* and *Java Server Pages TM (JSP).*

Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more generic approach in creating web pages. JSP technology provides an extensible way of generating dynamic content for a Web client.

4.3.1.1.2 Service Technologies

The J2EETM platform [SUN02] service technologies allow applications to access variety of services. The popular service technologies supported are *JDBCTM API* [SUN03a] which provides access to databases, *Java Transaction API (JTA)* [SUN03b] for transaction processing, *Java Naming and Directory InterfaceTM (JNDI)* [SUN03c] which provides access to naming and directory services, *J2EETM Connector Architecture* [SUN03d] which supports access to enterprise information systems and *Java API for XML Processing (JAXP)* [SUN03e] which enables applications to parse and transform XML documents independent of a particular XML processing implementation**.** The JDBCTM API 2.0 service technology is used in this prototype; a discussion of it is given below.

4.3.1.1.2.1 JDBC^{TM} API 2.0

The JDBC^{TM} API provides methods to invoke SQL commands from Java programming language methods. The JDBC API has two parts: an application-level interface used by the application components to access a database, and a service provider interface to attach a JDBC driver to the J2EE^{TM} platform.

4.3.1.1.3 Communication Technologies

Communication technologies provide a mechanism for communication between clients and serves and between the collaborating objects hosted by different servers. Some of the communication technologies supported by the J2EE^{TM} platform include, Transport Control Protocol over Internet Protocol (TCP/IP), Hypertext Transfer Protocol HTTP 1.0, Secure Socket Layer SSL 3.0, Java Remote Method Protocol (JRMP), Java IDL, Remote Method Invocation over Internet Inter ORB Protocol (RMI-IIOP), Java Message Service 1.0 (JMS), JavaMail and Java Activation Framework. The current prototype uses the HTTP 1.0 Protocol for communication between the browser-based clients (system administrator and system integrator) and the server-side entities. The components communicate among themselves on the server-side using Java Remote Method Invocation (RMI). The following subsection provides an overview of communication technologies used in this prototype.

4.3.1.1.3.1 HTTP 1.0 Protocol

The *Hypertext Transfer Protocol* (HTTP) [WWW03] is an application-level protocol for distributed, collaborative, hypermedia information systems (used by the World Wide Web). It is a generic stateless protocol, featuring the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

4.3.1.1.3.2 Java Remote Method Invocation (RMI)

Java RMI [SUN03f] is a set of API in Java that enables programmers to develop distributed applications. It uses Java language interfaces to define remote objects, and it combines Java serialization technology and the Java Remote Method Protocol (JRMP) for performing remote method invocations.

4.3.1.2 Web Servers and Application Servers

A *Web Server* consists of hardware and software programs, which serve the requested HTML pages or files (such as images, documents, etc) to the remote clients via web-browsers. Web servers are assigned IP addresses which remote applications can use to access them with the HTTP protocol. Apache HTTP Server [APA04], IBM HTTP Server [IBMa] and Sun Java System web server [SUN] are some of the popular examples of web servers.

Application Servers are software components that provide the business logic for an application program. They coordinate with web servers to return specific results (dynamic results) to a client's request. The clients communicate with the web servers using HTTP protocol and they in-turn communicate with the application servers to retrieve dynamic content, the web server return this information back to the clients using HTTP protocol. The application servers communicate with back-end database systems/directory servers using standard communication protocols. Some examples of application servers are IBM WebSphere Application Server [IBMb], BEA's WebLogic Server [BEA] and Internet Information Services (IIS) [IIS].

4.4 URDSMMS Implementation

Figure 4.5 illustrates the implementation of URDSMMS. The architecture, along with the existing URDS, is a multi-tier implementation of a distributed application model, which implies that various parts of the system can run on different remote machines. This architecture is complaint with the J2EE^{TM} distributed application model.

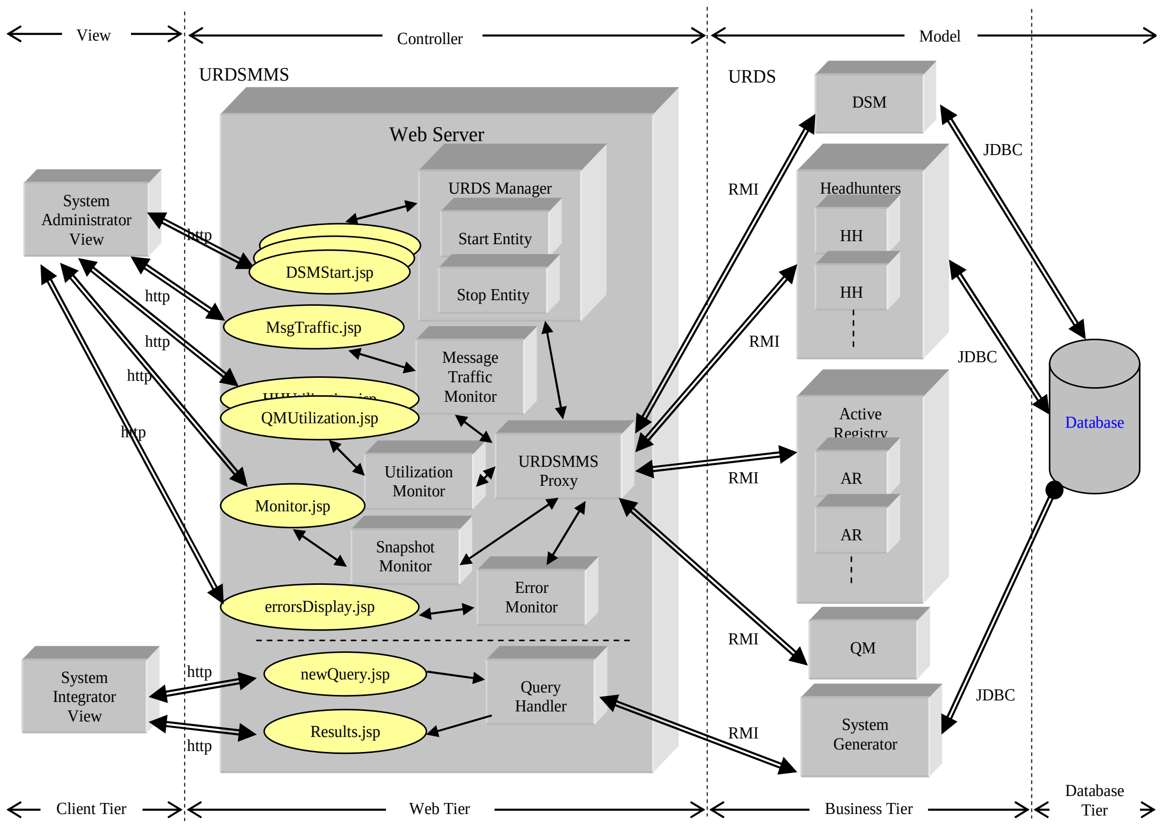


Figure 4.5 URDSMMS Implementation

The URDSMMS functionality is partitioned into modules, with each module divided into specific objects, each having a separate functionality. The prototype adopts the Model-View-Controller (MVC) architecture [GAM95, SUN02a]. The MVC architecture can be described as follows: “The *Model* represents the application data and the rules that govern the access and modification of this data. The *View* renders the contents of a model. It accesses data from the model and defines how that data should be presented. The *Controller* defines the application behavior; it translates user gestures into actions to be performed by the model” [GAM95].

The *Model* consists of the Business and Database tiers. It includes the underlying URDS system implemented in [SIR03]. The Business tier, which consists of different entities of

URDS, has been modified, so as to support the monitoring features of URDSMMS. The Oracle database tier includes the relational database tables of URDS and URDSMMS.

The *View* consists of browsers, which provide interfaces for the users (system administrator and system integrator) of URDSMMS. These users interact with the system and together form the client tier of URDSMMS.

The *Controller* of URDSMMS consists of JSP pages and proxy classes. The proxy classes are responsible for arbitrating between the JSP pages and entities of URDS in the Business tier. The JSP pages receive inputs forms the users (administrator and integrator) of the Client tier and send back the processed results via browsers.

4.4.1 Platform and Environment

In the prototype, the components of URDSMMS are implemented using Java^{TM} 2 Platform, Standard Edition (J2SE) [SUN03] version 1.4.1 software environment. The core entities are implemented as Java-RMI based services. The database oriented implementations are based on Oracle version 9.2. The web-based components (JSPs), which service client (system administrator and system integrator) interactions, are housed in the Tomcat 3.1.1 Servlet/JSP Container [APA03].

4.4.2 Communication Infrastructure

The interactions between the users (system administrator or system integrator) and the components in the Web tier are established using HTTP protocol. The communication between proxy classes in the Web tier and the URDS entities of the Business tier are based on Java RMI. The connections to the databases are established using the JDBC APIs.

4.4.3 Implementation Details

To structure an application along the lines of MVC architecture it is necessary to divide the application into objects and assign these objects to different tiers. This process is referred to as object decomposition. These are objects that deal with presentation aspects of the application, objects that deal with the application rules and data, and objects that accept and interpret user requests and control the application objects to fulfill these request [SUN01a].

The URDSMMS, when combined with the URDS, forms a four-tier architecture. The description in this section is presented in the following order: *Client Tier*, *Web Tier*, *Business Tier* and *Database Tier*.

4.4.3.1 Client Tier

The client tier consists of the users (system administrator and system integrator) of the URDSMMS. This client tier forms the view of the MVC architecture. It determines the presentation of the user interface of the prototype. The JSP pages work along with the components of URDSMMS and URDS. The JSP pages are used for dynamic generation of HTML responses.

As indicated earlier, this prototype considers two kinds of users, and hence, two views have been implemented, the system administrator view and system integrator view. Both the users login into the system securely by entering the username and password in the *main.jsp* page. A description about the JSP pages which present views to both the user are presented in section 4.4.3.2.1.

The Figure 4.6 illustrates the login page (main.jsp) of the URDSMMS, it serves as a common login page for both kinds of users.

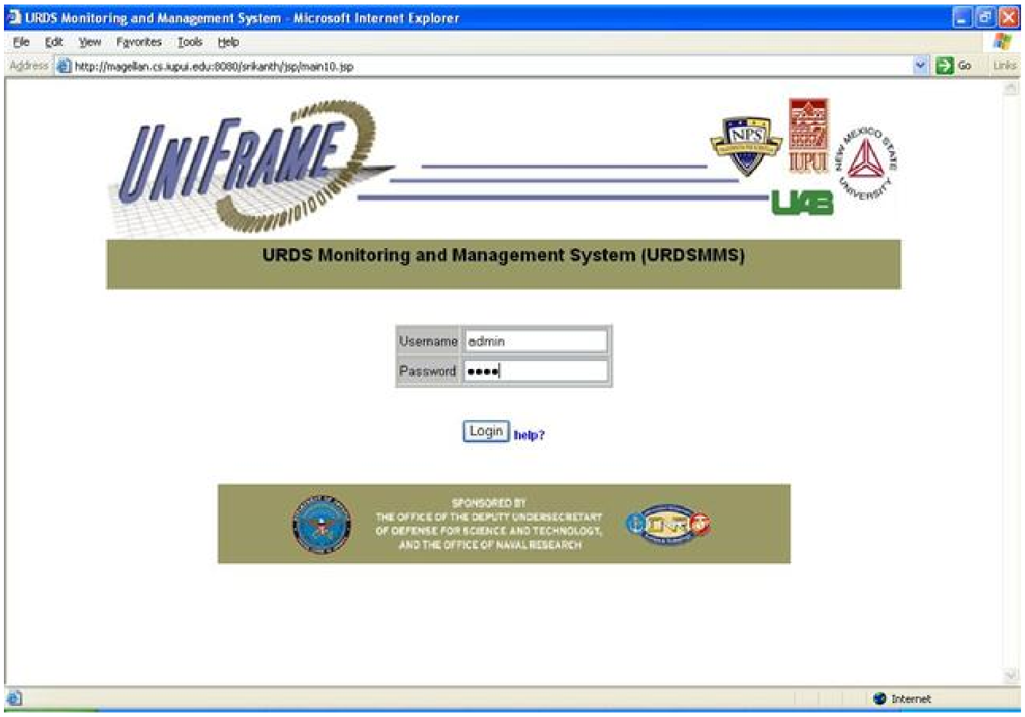

URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://magellan.cs.iupui.edu:8080/srikanth/jsp/main10.jsp
UNIFRAME
URDS Monitoring and Management System (URDSMMS)
Username admin
Password ••••
Login help?
SPONSORED BY THE OFFICE OF THE DEPUTY UNDERSECRETARY OF DEFENSE FOR SCIENCE AND TECHNOLOGY, AND THE OFFICE OF NAVAL RESEARCH
Internet

Figure 4.6 main.jsp view

4.4.3.2 Web Tier

The Web tier forms the controller module of the MVC architecture. It is responsible for coordinating the model and view. The Web Tier in the URDSMMS also consists of several proxy classes, which help the connection between Web Components in this tier and Business Components in the Business Tier.

4.4.3.2.1 Web Components

The Web Components consist of JSP pages, which interact with the proxy classes and return the results to the users.

main.jsp: This JSP page provides the login view for accessing the system. A user (system administrator/system integrator) enters the username and password to login into the URDSMMS system. If it is a successful login, the user is redirected to an authorized page (depending on system administrator view or system integrator view), else an error message is displayed.

DSMStart.jsp: This JSP page provides the view for initializing (starting) the DomainSecurityManager (DSM). The DSM can be hosted on any remote host.

HHStart.jsp: This JSP page provides the view for initializing (starting) a Headhunter (HH). The HH can be hosted on any desired remote host, by providing the necessary username, password and domain name, along with the DSM’s location.

ARStart.jsp: This JSP page provides the view for initializing (starting) a ActiveRegistry (AR). The AR can be hosted on any desired remote host, by providing the necessary username, password and domain name, along with the DSM’s location.

QMStart.jsp: This JSP page provides the view for initializing (starting) a QueryManager (QM). The QM can be hosted on any desired remote host, by providing the necessary username and password, along with the DSM’s location.

Monitor.jsp: This JSP page provides the view for displaying the statistical details of the URDS, such as location where the DSM, QM, HHs and ARs are running. An option to display details specific to each HH and AR (such as HH’s or AR’s name, domain, along with a feedback for each HH) is also provided. Number of queries the system is currently executed is also displayed.

getTraffic.jsp: This JSP page provides the view to display the message traffic, upon the request from the user. The message traffic is displayed in three categories, namely, authentication, query processing and component messages it gathers information from the *MsgTraffic* class.

graphViewer.jsp: This JSP page provides the view to display the response times for executed queries. The details are shown graphically, for the latest 20 queries, so as to provide better understanding of the system.

HHUtilization.jsp: This JSP page provides the graphical view of the utilization of a Headhunter. A graph of the latest 100 values can be plotted by selecting the HH for which the graph is required. This information is obtained form the *Monitor* class.

QMUtilization.jsp: This JSP page provides the graphical view of the utilization of the Query Manager. A graph of the latest 100 values can be plotted upon the user's request. This information is obtained form the *Monitor* class.

DSMStop.jsp: This JSP page provides the view for killing a Domain Security Manager. This process can be achieved by submitting the location of the DSM to the *Monitor* class.

QMStop.jsp: This JSP page provides the view for killing a Query Manager. This process can be achieved by submitting the location of the QM to the *Monitor* class.

HHStop.jsp: This JSP page provides the view for killing a Headhunter. This process can be achieved by submitting the location of the HH to the *Monitor* class.

ARStop.jsp: This JSP page provides the view for killing a Active Registry. This process can be achieved by submitting the location of the AR to the *Monitor* class.

Help.htm: This HTML page provides a navigational help for the administrator view of the entire system and also provides help for the known error messages.

errorsDisplay.jsp: This JSP page serves as a view for displaying all the error messages that occur in the URDS.

newQuery.jsp: This JSP page serves as a view for submitting a query for processing to the *URDSProxy* class.

displayResults.jsp: This JSP page is to view the processed results. The results can be viewed by submitting queryid.

Help_si.htm: This HTML page provides a navigational help for the system integrator view.

The *newQuery.jsp* and *displayResults.jsp* pages provide interfaces for the system integrator view while the rest of the JSP pages serve as interfaces for the system administrator view. The screen shots for these views are provided in appendix C.

4.4.3.2.2 URDSMMS Classes

These classes serve as connectors between the Web tier and Business tier. These Java classes collect monitoring information from different entities of URDS. The functionality of these classes of URDSMMS can be described as below and the class diagrams are provided in appendix E.

URDS Management: This connects the *DSMStart.jsp, HHStart.jsp, HHStop.jsp,* etc. (other JSP pages which fall-in the management activity of URDSMMS) in the Web tier with the URDS entities (DSM, HH, AR &QM) in Business tier.

Message Traffic Monitor: This class connects *getTraffic.jsp* in the Web tier with all URDS entities (DSM, HH, AR and QM) in Business tier. It records the traffic information by placing probes in the URDS entities called software probes [MAN95].

Utilization Monitor: This class connects *HHUtlization.jsp* and *QMUtilization.jsp* in Web tier to HHs and QM in the Business tier. It records the net utilization of HHs and QM at regular time intervals.

Snapshot Monitor: This class connects *Monitor.jsp* in Web tier to entities in URDS (DSM, HH, AR &QM) and, inturn, connects to the database to collect the statistical details.

Query Handler: This class connects *newQuery.jsp* (accepts queries) and *displayResults.jsp* in Web tier to System Generator in the Business tier.

4.4.3.3 Business Tier

The Business tier consists of business components, which forms a part of Model in the MVC architecture. Business Components here refer to standalone software entities that provide services to components in other tiers or in the same tier. These business components can be remotely accessed using standalone communication protocols. The business components in URDS consist of DSM, HH, AR and QM.

4.4.3.4 Database Tier

The Database tier is responsible for storing persistent data of URDSMMS and URDS. The database is an Oracle database (version 9.2) which is a relational database. The database is accessed and updated through the JDBC technology.

This chapter provided an overview of the URDSMMS architecture and an explanation about each of its entities. The use-case diagrams for the design of UI and the implementation details of URDSMMS are also discussed in this chapter. The next chapter will discuss the experimentation of URDSMMS and the analysis of the results.

5. EXPERIMENTATION AND VALIDATION

The chapter 4 provided the design and implementation details of URDSMMS. This chapter focuses on the experimental analysis and validation of the prototype created. The usability of the user interface of the prototype is validated based on an informal survey.

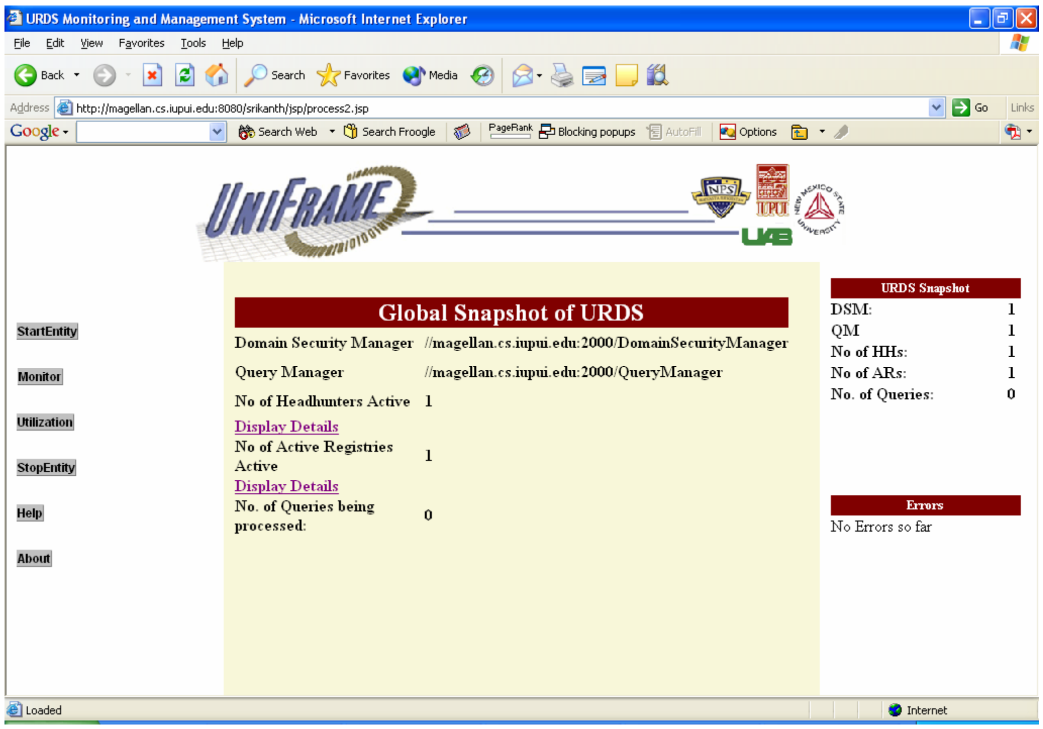
5.1 Experimental Setup

The prototype was implemented using the Java^{TM} 2 Platform, Standard Editon (J2SE) version 1.4.1 software environment. The core entities (URDS Management, Message Traffic Monitor, Utilization Monitor, URDS Snapshot Monitor and Error Monitor) were implemented using Java and Java-RMI technologies. The user interface was designed using JSP; and deployed on Jakarta Tomcat version 3.1.1. The underlying URDS system was adapted from [SIR02] and [DEV05]. The repository used by the Domain Security Manager for authenticating other entities was implemented using Oracle9i database version 9.2. URDSMMS also used the same database to store results. The hardware platform used for the experimentation was Sun Solaris Ultra-250 Sparc machines, which are running with Sun OS release 5.8. The prototype can be implemented on any environment using J2SE 1.4.1, or later version, and running any web server which is compatible with JSP technology.

5.2 Experiments to empirically validate the Monitoring System

URDSMMS was tested with different sets of experiments to empirically validate the functioning of the monitoring system. All the experiments were conducted on the version of URDS which has been implemented with the multi-threaded Query Manager and single-threaded Headhunters [DEV05]. This version of URDS was capable of processing multiple queries simultaneously.

This set of experiments was carried out with one Headhunter (HH) and one Active Registry, with the utilization of Headhunter and Query Manager being calculated at regular time intervals. After a period of time, two queries were submitted to the system for processing, to trace the appropriate changes in the utilization values. The reason for considering such a small configuration is to clearly observe the change of pattern in the graphs. Figure 5.1 illustrates the Snapshot of URDS; it depicts the number of active entities of URDS.


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp Go Links
Google Search Web Search Froogle PageRank Blocking popups AutoFill Options
UNIFRAME
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

Global Snapshot of URDS

Domain Security Manager	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Query Manager	//magellan.cs.iupui.edu:2000/QueryManager
No of Headhunters Active	1
Display Details	
No of Active Registries Active	1
Display Details	
No. of Queries being processed:	0

URDS Snapshot

DSM:	1
QM	1
No of HHs:	1
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure 5.1 Snapshot of URDS

In Figure 5.2, the vertical represents the utilization percentage of a Headhunter (Headhunter's active time) and the horizontal represents the latest calculations of the utilization (history of values up to 100 are maintained). From the graph, the utilization of a Headhunter is observed to be in the range of 0.8% and 1.5% when it is idle, i.e., no queries are being sent to that Headhunter for processing. This small percentage of utilization is accounted to the Headhunter's multicasting and receiving the updated component information from the Active Registry. The peak value of 7.0% utilization is because the Headhunter receives two queries from the Query Manager for processing. If the query input pattern was continuous, then this peak value of utilization might have been continued for a longer period (as long as the Headhunter keeps processing new queries). Figure 5.2 shows the utilization of Headhunter for the above described scenario.

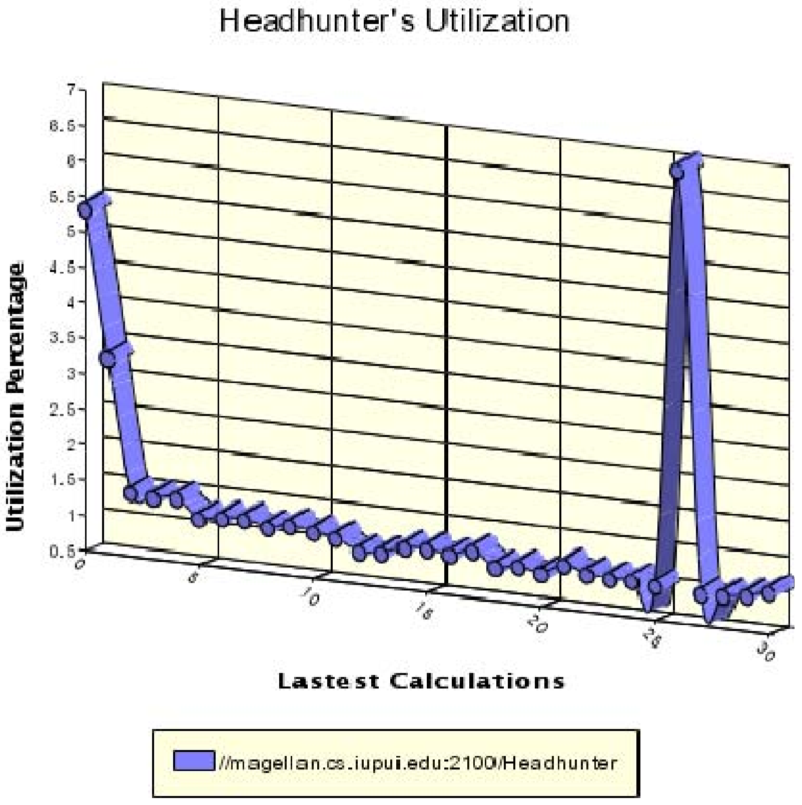


Figure 5.2 Utilization of Headhunter

Figure 5.3, illustrates the utilization of the Query Manager. The vertical represents the utilization percentage of the Query Manager and the horizontal represents the latest calculations of the utilization (history of values up to 100 are maintained). The utilization of the Query Manager is approximately 0.01% when it is idle (Query Manager is not receiving any queries from the system integrators). This small percentage is accounted to the Query Manager checking in its query queue (queue to store the incoming queries). A peak value of 83% utilization is seen at a point when the Query Manager received two queries from the system integrator for processing. In the case of continuous query input pattern, the utilization value would have been a peak value for a longer period (as long as the Query Manager keeps receiving new queries from system integrators), the value depends upon the rate at which the queries are being supplied to it.

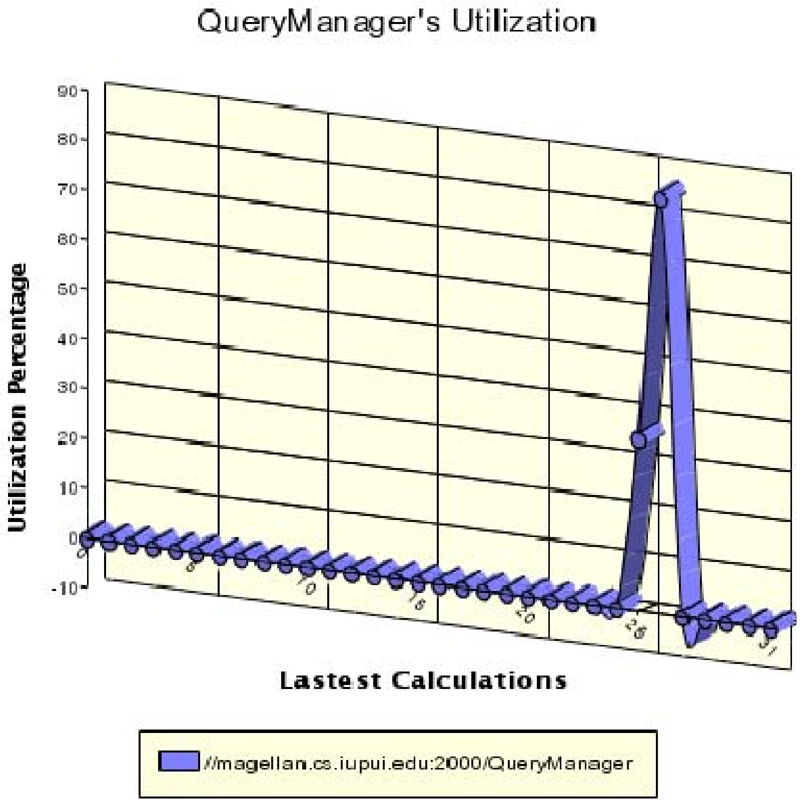


Figure 5.3 Utilization of Query Manager

5.3 Experiments to Test Scalability

These set of experiments were carried out to observe the scalability of the prototype. The prototype was tested with 20 Headhunters running across 20 Sun Ultra-250 Sparc machines and 4 Active Registries running on 2 machines. The utilization of each Headhunter and the Query Manager was recorded and displayed.

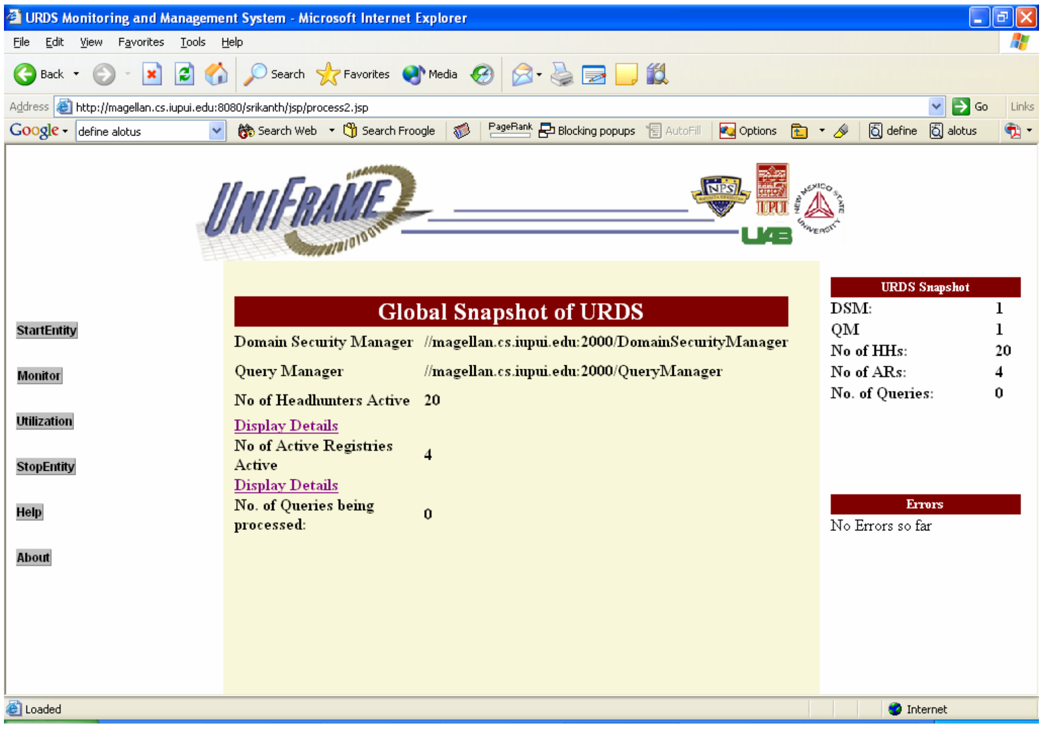
Table 5.1 shows the message traffic of URDS when 20 Headhunters are running and 4 Active Registries running. Since no new entities are initiated and no new queries are being submitted the number of *Authentication Messages* and *Query Processing Messages* are 0, respectively. The number of messages that account for Component Registration is

4; this is because of the Headhunter multicasting its location and the Active Registry responding back with the updated components list.

Classification	No. of Messages
Authentication Messages	0
Query Processing Messages	0
Component Registration	4

Table 5.1 Message Traffic of URDS

Figure 5.4 illustrates the snapshot of the URDS with 20 active Headhunters, 4 Active Registries, and one Domain Security Manager and one Query Manager. By clicking on the display details of the Headhunter or the Active Registry, the system integrator can view the specific details of each Headhunter or Active Registry. This view helps to know the statistical details about the entities of URDS and their distribution on different remote hosts.


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

Global Snapshot of URDS

Domain Security Manager	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Query Manager	//magellan.cs.iupui.edu:2000/QueryManager
No of Headhunters Active	20
Display Details	
No of Active Registries Active	4
Display Details	
No. of Queries being processed:	0

URDS Snapshot

DSM:	1
QM	1
No of HHs:	20
No of ARs:	4
No. of Queries:	0

Errors

No Errors so far

Figure 5.4 Snapshot of URDS (showing 20 HHs)

5.4 Experiments to test the functionality of URDSMMS

The following experiments were carried out to test the features provided by the URDSMMS. The URDSMMS features assist the system administrator in making appropriate decisions to improve the performance of the URDS.

Experiment to reduce average response time by adding Headhunters: The goal of this experiment is to show that, by adding new Headhunters to the URDS, average response time reduces. To demonstrate this, a graph is plotted between *average response time* and *system time* (time from which Query Manager starts receiving queries from system integrator). Figure 5.5 illustrates a graph, comparing the response times for 4 series of values, each set of values were recorded when the URDS was running with 2HHs, 4HHs, 6HHs and 8HHs respectively.

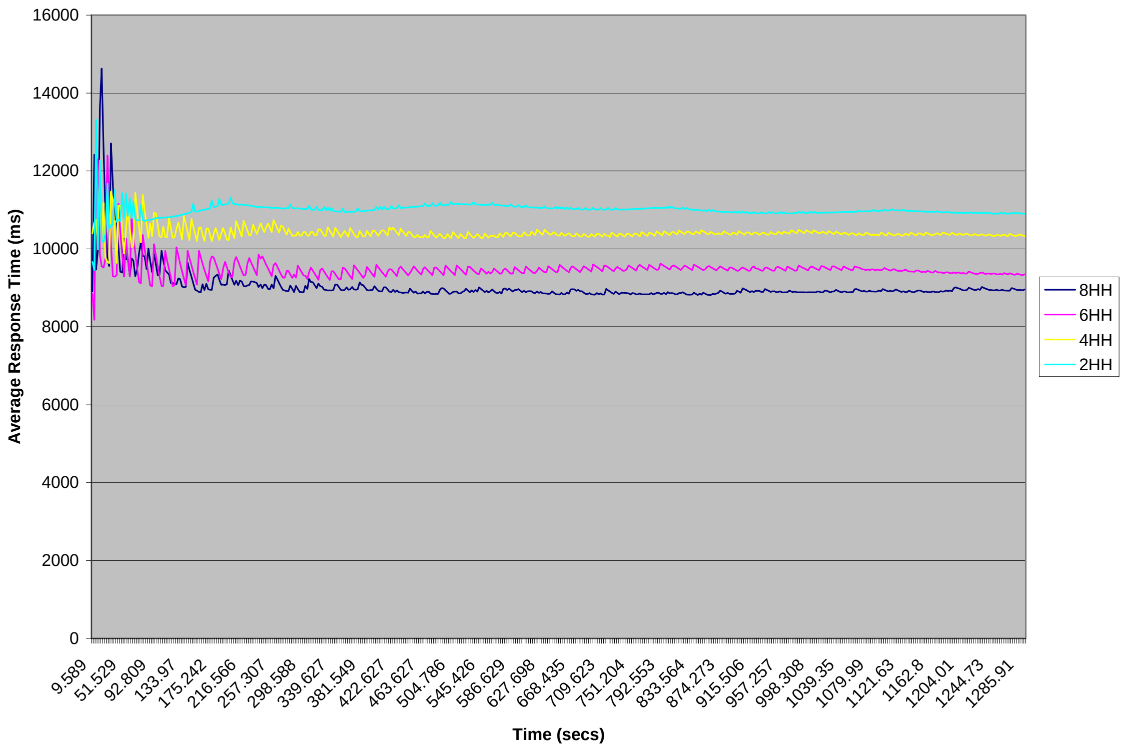


Figure 5.5 Comparison of average response times for different sets of Headhunters

All the 4 series values were recorded under similar condition, with queries being sent from three system integrators at a rate of 1 query for every 2.5 seconds (these numbers are chosen randomly to carry out the experiment). To measure the performance of the system, the same query is issued repeatedly.

From the graph we can observe that initially the response time values are high (for a few instances). One of the reasons for these initial peak values could be the time to access the data structure (in this case it is a Hashtable). Since the same data structure is being accessed multiple number of times (which increases the hit rate for that data structure), the operating system stores the result in its cache, because of which for the further queries the response time slowly reduces and reaches a constant state. An expanded version of the same graph is shown in Figure 5.6. The decrease in the average response time is seen because, by adding new Headhunters, more number of queries can be processed simultaneously and thereby reducing the waiting time for each query (in Query Manager's query queue).

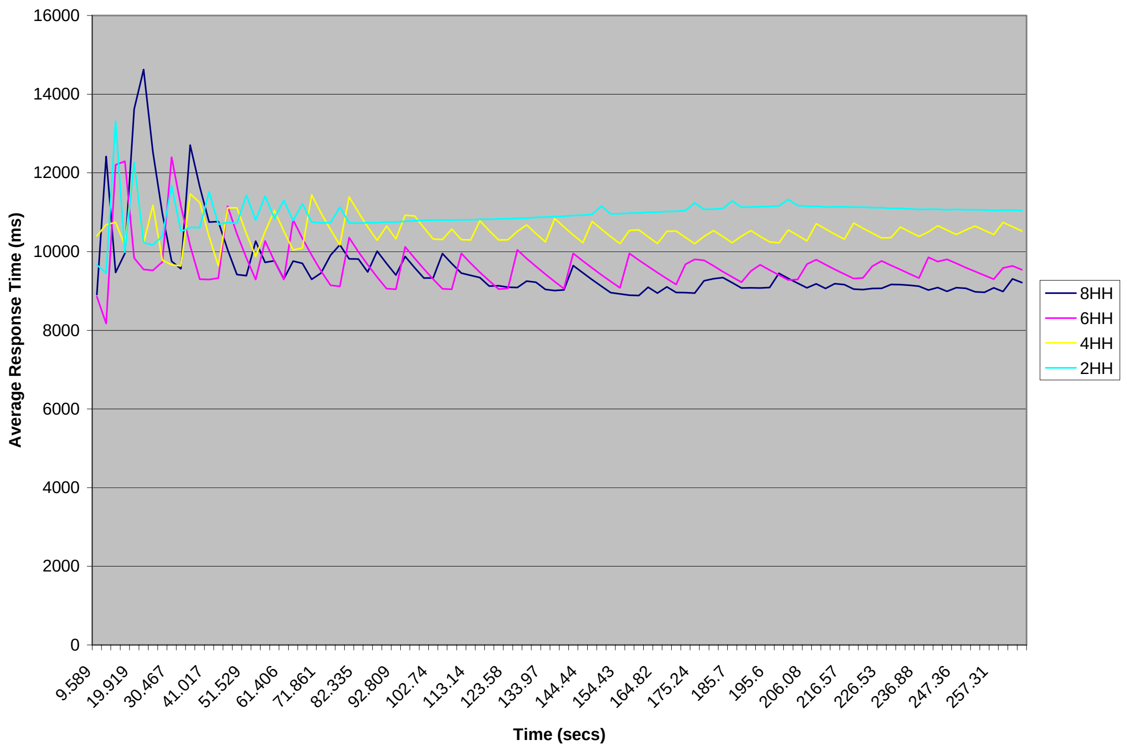


Figure 5.6 Comparison of average response times for different sets of Headhunters (enlarged version)

Experiment to increase the throughput by adding Headhunters: The aim of this experiment is to show that by adding new Headhunters to the URDS, a greater number of queries can be processed within less time, thereby, increasing the throughput of URDS. To illustrate this behavior a graph is plotted between *time* (time from which Query Manager starts receiving queries) and *number of queries processed*.

Figure 5.7 illustrates such a graph, comparing the number of queries processed at certain times. The graph shows 3 series of values, each set of values was recorded when the URDS was running with 2HHs, 4HHs and 8HHs respectively.

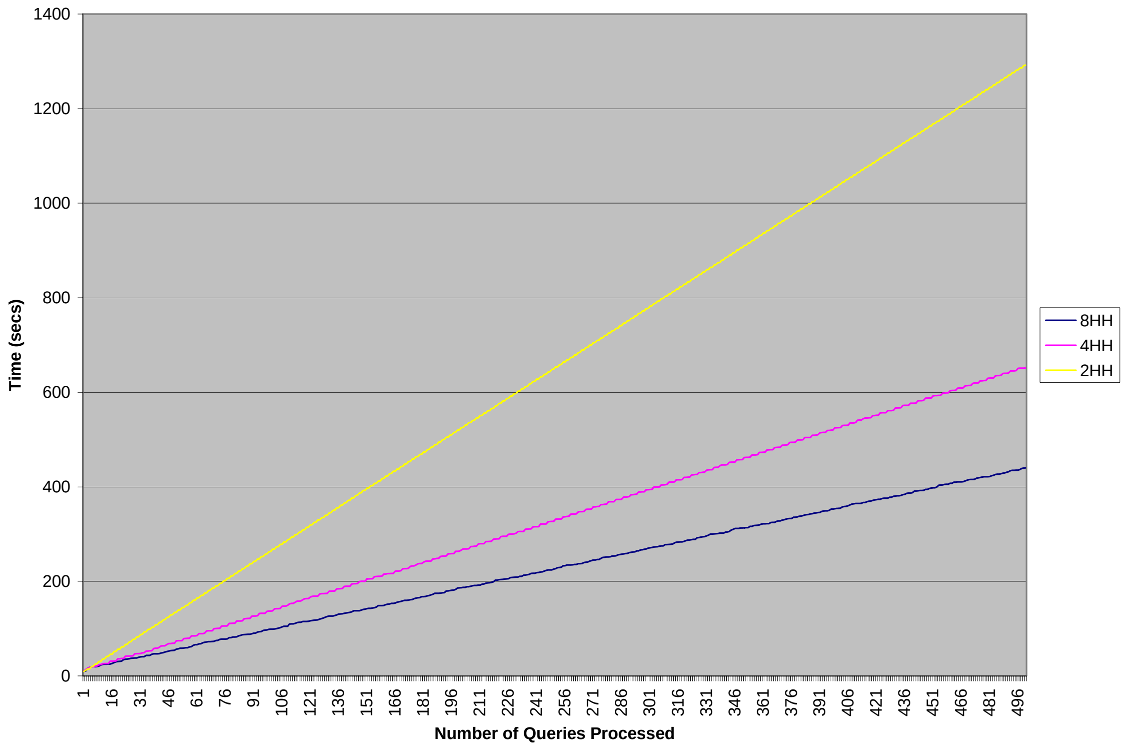
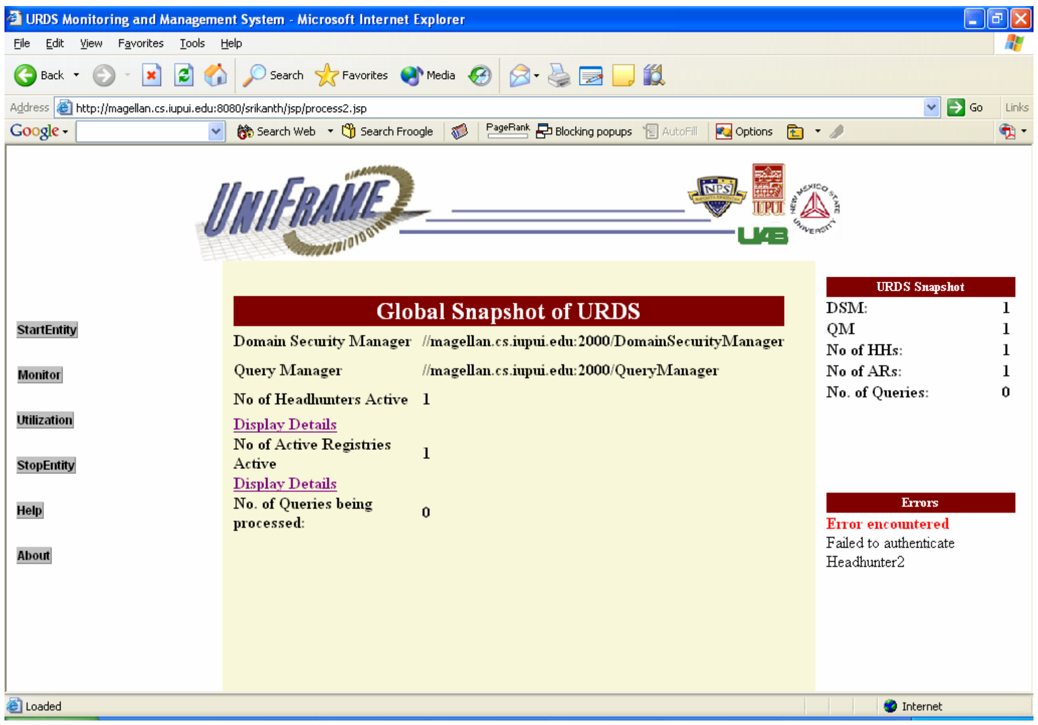


Figure 5.7 Comparison of number of queries processed for different sets of Headhunters

From the graph it is observed that, as a greater number of Headhunters are added to the URDS, the slope of the graph further reduces. This decrease in slope is expected to happen until the point at which the query arrival rate is almost the same as the results are processed and returned back to the system developer, i.e., there will be no waiting queries in the Query Manager's query queue. At this saturation point the slope will not further decrease by adding new HHs as there will be no waiting queries to process. The observations made from the graph are as follows; at 200 seconds for a 2HHs scenario only 76 queries were processed, where as in the 4HHs and 8HHs cases, 150 and 220 queries were processed respectively.

View showing the error handling feature of URDSMMS: Figure 5.8 illustrates a screenshot of the URDSMMS when a Headhunter attempts to login with an incorrect username/password. Static help messages for common (known) error messages help the system administrator in taking the corrective actions to recover from the errors. In this particular case the administrator is advised to re-instantiate that particular Headhunter.


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
Google Search Web Search Froogle PageRank Blocking popups AutoFill Options
UNIFRAME
NPS
IUPUI
UAB
New Mexico State University
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

Global Snapshot of URDS

Domain Security Manager	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Query Manager	//magellan.cs.iupui.edu:2000/QueryManager
No of Headhunters Active	1
Display Details	
No of Active Registries Active	1
Display Details	
No. of Queries being processed:	0

URDS Snapshot

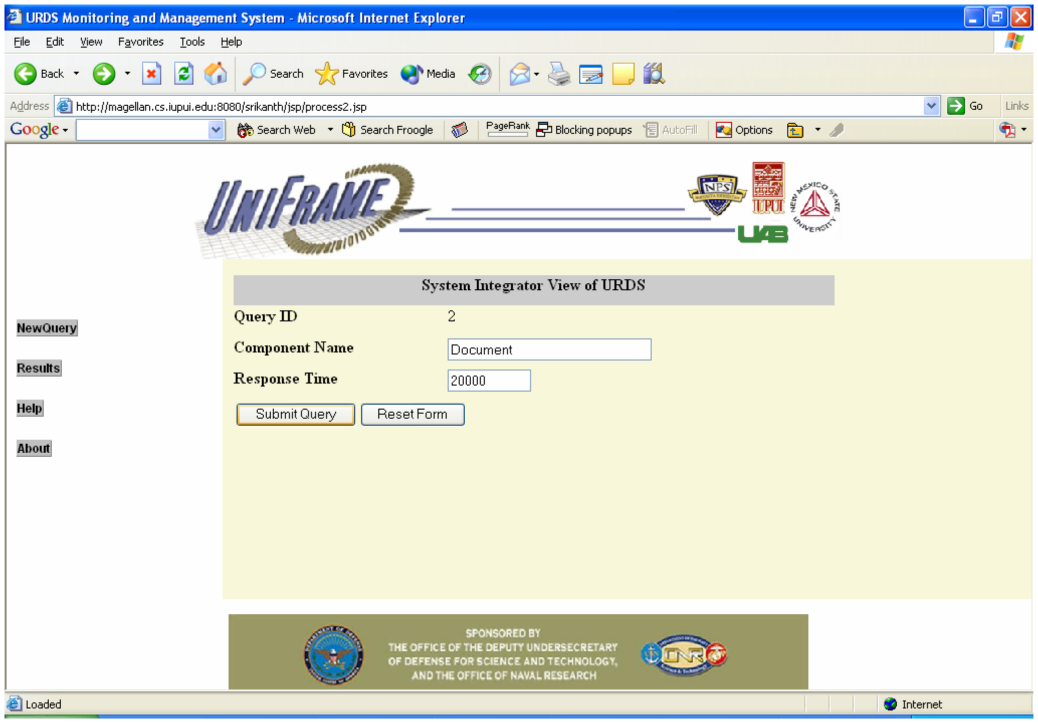
DSM:	1
QM	1
No of HHs:	1
No of ARs:	1
No. of Queries:	0

Errors

Error encountered
Failed to authenticate Headhunter2

Figure 5.8 View showing error handling feature

View showing the submit query feature of URDSMMS: Figure 5.9 illustrates a screenshot of the URDSMMS when a system integrator tries to submit a new query for processing. The unique query id is automatically generated; the system integrator needs to enter the component name (or domain name, the present example is illustrated for domain name) and the response time for retrieving the results.

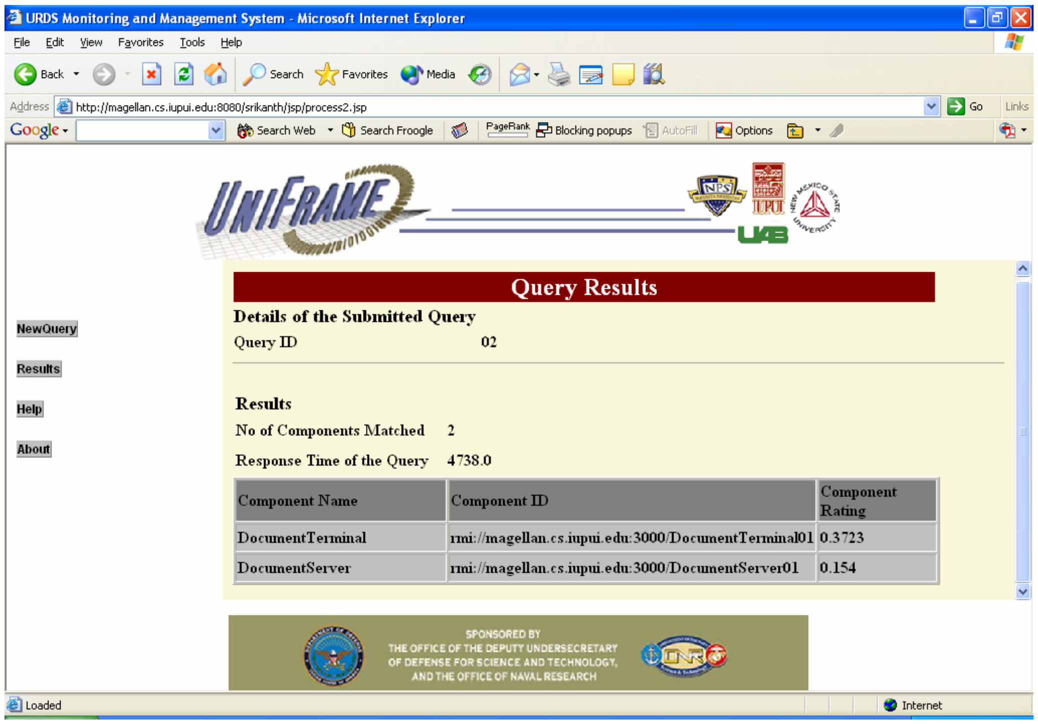

URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
Search Web Search Froogle PageRank Blocking popups AutoFill Options
UNIFRAME
NewQuery
Results
Help
About
Submit Query
Reset Form
SPONSORED BY THE OFFICE OF THE DEPUTY UNDERSECRETARY OF DEFENSE FOR SCIENCE AND TECHNOLOGY, AND THE OFFICE OF NAVAL RESEARCH
Loaded
Internet

System Integrator View of URDS

Query ID	2
Component Name	Document
Response Time	20000

Figure 5.9 View showing the submit query feature

View showing the retrieve results feature of URDSMMS: Figure 5.10 illustrates a screenshot of the URDSMMS, when a system integrator tries to retrieve results for a particular query id. This view shows the details (such as number of components matched and response time) of the submitted query and the details (such as component name, component ID and its rating) for each component.


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
Google Search Web Search Froogle PageRank Blocking popups AutoFill Options
UNIFRAME
NPS IUPUI UAB New Mexico State University
NewQuery
Results
Help
About
SPONSORED BY THE OFFICE OF THE DEPUTY UNDERSECRETARY OF DEFENSE FOR SCIENCE AND TECHNOLOGY, AND THE OFFICE OF NAVAL RESEARCH
Loaded
Internet

Query Results

Details of the Submitted Query

Query ID 02

Results

No of Components Matched 2

Response Time of the Query 4738.0

Component Name	Component ID	Component Rating
DocumentTerminal	rmi://magellan.cs.iupui.edu:3000/DocumentTerminal01	0.3723
DocumentServer	rmi://magellan.cs.iupui.edu:3000/DocumentServer01	0.154

Figure 5.10 View showing the retrieve results feature

The screen shots of other features of the URDSMMS with both system administrator and system integrator views are provided in Appendix C.

The experiments in this subsection were performed to test the functionality of the URDSMMS. From the sets of experiments, it can be observed that the features are helpful in making managerial decisions to improve the throughput and average response time of the URDS. The features also show the error handling capability of the URDSMMS.

5.5 Usability testing of the interface

“Usability testing is a highly efficient method for measuring the quality of use and acceptance of a product, and detecting issues” [VNE03]. The purpose of usability testing is to ensure that the monitoring system will meet the needs of the target user group. In the case of URDSMMS, the testing involved the observation of the users performing real tasks with the URDSMMS, recording what they do, analyzing the results, and making appropriate changes. This subsection presents how some of the usability goals of [PRE02] were tested and addressed in the URDSMMS.

After the system has been fully implemented, a survey was prepared on the basis of [PRE02], to collect the user's experiences. The UniFrame team members were requested to use the URDSMMS and were asked to fill a survey, to collect their feedback about the system.

The survey consisted of questions for obtaining suitable feedback about the effectiveness, utility, learnability and memorability [PRE02] of the interface. More explanation about these metrics, consequent questions and the results are included in Appendix D. The survey also included questions aiming at measuring user experience goals [PRE02], such as user's satisfaction and degree of aesthetically pleasing. Some of the user suggestions from the survey included redesigning the menu buttons and reducing header and footer size. These suggestions were suitably incorporated, resulting in an aesthetically pleasing user interface for the URDSMMS. Also, based on the suggestions made by the users, appropriate help was provided for error messages.

This chapter discusses the experimental results and validation of the deployed monitoring and management system. It provides different graphs and the analysis of those graphs, validating the features of URDSMMS. In addition by using the survey, the usability of the interface has been verified. The next chapter summarizes this report by presenting the conclusions, possible future work and a summary of the report.

6. CONCLUSION AND FUTURE WORK

This project report presented a framework and an implementation of a monitoring and management prototype for the URDS. The present chapter discusses the conclusion, overview of the features if the URDSMMS, future work and a brief summary about this research work.

6.1 Conclusion

In this project, the features of a monitoring and management system suitable for software component discovery systems were explored and these features were implemented for the URDS system. This work was primarily motivated by the lack of a monitoring and monitoring system that satisfies the requirements of component based discovery systems (discussed in chapter 3).

The following are the conclusions that can be drawn by the implementation of URDSMMS and from the experimental results shown in the previous chapters:

- The information displayed by the URDSMMS is in accordance with the data produced by the underlying system (URDS).
- The information displayed by the URDSMMS, such as utilization of entities (QM and HH), average response time, etc., helps the system administrator in determining the overall performance of the system.
- A system administrator can control and coordinate the entities of URDS, by instantiate and terminate entities and also monitor the system behavior.
- The interface of URDSMMS improves the usability of the system and serves as an easy-to-use interface for a novice administrator/integrator, to oversee the functionality of the system.

6.2 Features of URDSMMS

- Provides a remote control (initiation and termination) of the entities in the URDS.
- Generates the utilization graphs for Headhunters and Query Manager, so as to reveal the performance of the system.
- Displays a continuous snapshot of the URDS and, optionally displays the details of the individual entities.
- Provides an option for system integrators to submit queries for processing and retrieve the results.
- Provides an error dialogue for the system administrator.

6.3 Future Work

Some of the possible future enhancements are:

- *Heterogeneous Environments:* Distributed system span heterogeneous technological platforms (such as .net, CORBA, etc). The present URDSMMS architecture has only been implemented for the J2EE platform; this can be extended to other heterogeneous platforms.
- *Hardware Resources:* The present monitoring framework only considers software resources. If hardware resources [MDS2, MAS04, HAW] (such as CPU utilization, memory, etc.) were also considered, it would help the system administrator to make better judgments, about where to deploy the components (depending on the available resources on a particular system).
- *Suspending Processes:* In the present architecture, if an entity (e.g. Headhunter) is idle for a long time, the system administrator can make the decision to kill that process and redeploy it on another remote host if necessary. Instead of this approach, the process can be suspended and redeployed on another remote host, thereby saving the present state of the process.
- *Error Monitoring Architecture:* Though the URDSMMS provides an error monitoring feature for the URDS, it does not consider catastrophic failures, such

as failure of a remote host or loss of messages, which would help in notifying such errors or failures.

- *Visualization of Monitoring Results:* Visualizing the monitoring information will enable the user to analyze the data easily. User interface toolkits like prefuse [PRE] can be used for building interactive visualizations. This enables data to be represented as a set of entities (or nodes) which can be connected by any number of relations (or edges).
- *Fault Handling Features:* The monitoring system does not take care of fault tolerance, e.g., if an error occurs, the system cannot handle the error dynamically, so it just notifies the system administrator regarding the error. The monitoring system should be able to provide an solution during runtime and therefore handle such errors.
- *Event Ordering:* In a DCS, an event can arrive out of order, which makes event ordering (partial or total) as an important issue to be considered in monitoring applications [LAM78]. For example, event traces may be merged or combined based on casual ordering. The monitoring system should be able to handle such an issue.

6.4 Summary

The research has presented a framework (URDSMMS) for monitoring and managing the URDS. This framework has been implemented, incorporating the key features for monitoring and by providing a user interface, which improves the usability of the URDS. The report also details the experiments that were performed to test the scalability and empirically validate and the features of the monitoring system. Thus, the URDSMMS architecture along with its interface makes it a promising solution for monitoring and management of the URDS.

LIST OF REFERENCES

[ALS98] Al-Shaer E., “A Hierarchical Filtering-based Monitoring Architecture for Large-scale Distributed Systems”, Ph.D. Dissertation, Computer Science Department, Old Dominion University, December 1998.

[AMI] The AMIDST project, see http://amidst.ctit.utwente.nl/

[ANJ04] Kumari A., “Synchronization and Quality of Service Specification and Matching of Software Components”, M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, December, 2004.

[APA03] Apache Jakarta Project. Apache Tomcat, 2003. http://jakarta.apache.org/tomcat/

[APA04] Apache HTTP Server Project, “Apache HTTP Server Version 1.3” http://httpd.apache.org/

[ARL2002] Arlow J., Neustadt I., “UML and the Unified: Process Practical Object-Oriented Analysis & Design”, Addison Wesley, 2002

[BEA] BEA WebLogic Server® 8.1 http://www.bea.com/products/weblogic/server/index.shtml

[BRA02] Brahnmath G., Raje R., Olson A., Bryant B., Auguston M., Burt C., Quality of Service Catalog for Software Components. *The Proceedings of the Southeastern Software Engineering Conference*, Huntsville, Alabama, USA, April 2002.

[CLA] ClassAd: http://www.cs.wisc.edu/condor/classad/index.html

[CON] Condor: http://www.cs.wisc.edu/condor/index.html

[COU01] Coulouris G., Dollimore J., Kindberg T., “Distributed System Concepts and Design”, Addison Wesley, 2001

[CZA00] Czarnecki K., Eisenecker U., *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.

[CZA01] Czajkowski K., et al., “Grid Information Services for Distributed Resource Sharing”. *10th IEEE International Symposium on High Performance Distributed Computing*. 2001: IEEE Press.

[DAT02] DataGrid Information and Monitoring Services Architecture: Design, Requirements and Evaluation Criteria”, Technical Report DataGrid-03-D3.2-334453-4-0 , DataGrid, 2002.

[DES] de Sousa M., Richardson D., “Issues on Software Monitoring”, Technical report, ICS, 2002.

[DEV05] Barun D., “Enhancement of UniFrame Resource Discovery Service”, M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, February, 2005.

[DIA00] Diakov N., Batteram H., Zandbelt H., van Sinderen M, “Monitoring of distributed component interactions”. *Proceedings of 7th Int. Conf. on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, Enschede, The Netherlands, October 2000.

[FIS01] Fisher S., “Relational Model for Information and Monitoring”. Technical Report GWD-Perf-7-1, GGF, 2001.

[FOS99] Foster I., Kesselman C.,“*The Grid: Blueprint of a New Computing Infrastructure*”, Morgan Kaufmann, 1999. ISBN 1-55860-475-8.

[FRI] The FRIENDS project, see http://friends.gigaport.nl/

[GAM95] Gamma E., Helm R., Johnson R., Vlissides J., “Design Patterns Elements of Reusable Object-Orientated Software”. Addison-Wesley, 1995.

[GAO00] Gao J, Zhu E., Shim S., Chang L., “Monitoring Software Components and Component-Based Software”. *Proceedings of the 24th Annual International Computer Software and Applications Conference*, COMPSAC 2000.

[GGF] Global Grid Forum: http://www.gridforum.org/

[GLO] Globus Toolkit : http://www-unix.globus.org/toolkit/

[HAW] Hawkeye: http://www.cs.wisc.edu/condor/hawkeye/

[HOF94] Hofmann R., Klar R., Mohr B., Quick A., Siegle M., “Distributed Performance Monitoring: Methods, Tools and Applications,” IEEE Transactions on Parallel and Distributed Systems, vol. 5, pp. 585{597, June 1994)

[IBMa] IBM, “HTTP Server version 2.0”.

http://www-3.ibm.com/software/webservers/httpservers/

[IBMb] IBM WebSphere® Application Server V6.0
http://www-306.ibm.com/software/webservers/appserv/was/

[IIS] Internet Information Services (IIS) 6.0
http://www.microsoft.com/WindowsServer2003/iis/default.mspx

[JOY87] Joyce J., Lomow G., Slind K., Unger B., Monitoring Distributed Systems. *ACM Transactions on Computer Systems*, vol. 5, no. 2, pp. 121-50, 1987.

[LAM78] Lamport L., "Time, clocks, and the ordering of events in a distributed system". *Communication ACM* 21, 7 (Jul. 1978), 558-565.

[LDA] OpenLdap: http://www.openldap.org/
[MAN95] Mansouri-Samani M., "Monitoring of Distributed Systems", Ph.D. Dissertation, Department of Computing, University of London, December 1995

[MAS04] Massie M., Chun B., and Culler D. "The Ganglia Distributed Monitoring System: Design, Implementation and Experience". *Parallel Computing*, April 2004.

[MDS2] MDS2: http://www.globus.org/mds/

[MYS04] Mysore P., Raje R., Olson A., Bryant B., Auguston M., Burt C.,"Scalability and fault handling issues in UniFrame Discovery Service," Technical Report TR-CIS-0705-04, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, 2004.

[OES99] Oestereich B., "Developing Software with UML Object-Oriented Analysis & Design in Practice", Addison Wesley, 1999

[PRE] prefuse: interactive information visualization, http://prefuse.sourceforge.net/

[PRE02] Preece J., Rogers Y., Sharp H., Interaction Design: beyond human-computer interaction, John Wiley & Sons, Inc., 2002.

[RAJ00] Raje R., UMM: Unified Meta-object Model. *Proceedings of 4th IEEE International Conference on Algorithms and Architecture for Parallel Processing*, Hong Kong, December 2000.

[RAJ01] Raje R., Bryant B., Auguston M., Olson A., Burt C., A Unified Approach for the Integration of Distributed Heterogeneous Software Components. *Proceedings of the 2001 Monterey Workshop on Engineering Automation for Software Intensive System Integration*, Monterey, California, USA, June 2001.

[SCH95] Schroeder B. “On-Line Monitoring: A Tutorial”, *IEEE Computer*, vol. 28, n. 6, June 1995, pp.72-77

[SIR02] Siram N., “An Architecture for Discovery of Heterogeneous Software Components.”, M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, March, 2002.

[SIR03] Siram N., Raje R., Olson A., Bryant B., Burt C., Auguston M, “An Architecture for the UniFrame Resource Discovery Service,” *Proceedings of SEM 2002, 3rd International Workshop on Software Engineering and Middleware*, May 20-21, 2002, Orlando, Florida Springer-Verlag Lecture Notes in Computer Science, Vol. 2596, 2003, pp. 20-35

[SOT02] Sottile M., Minnich R., “Supermon: A highspeed cluster monitoring system”. *Proceedings of Cluster*, September 2002.

[SUN] Sun Java System Web Server 6.1 http://www.sun.com/software/products/web_srvr/home_web_srvr.xml

[SUN01] Sun Microsystems. Java^{TM} 2 Platform Enterprise Edition Specification, Version 1.3. August 2001. http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf.

[SUN01a] Sun Microsystems. Designing Enterprise Applications with the J2EE Platform http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/platform_technologies/index.html, 2001

[SUN02] Sun Microsystems. The J2EE Tutorial^{TM}. April 2002. http://java.sun.com/j2ee/tutorial/1_3-fcs/.

[SUN02a] Java BluePrints - J2EE Patterns, 2002 http://java.sun.com/blueprints/patterns/MVC-detailed.html,

[SUN03] Sun Microsystems. Java 2 Platform, Standard Edition (J2SE), v1.4.0. 2003. http://java.sun.com/j2se/1.4/.

[SUN03a] Sun Microsystems. JDBC^{TM} API. 2003. http://java.sun.com/products/jdk/1.2/docs/guide/jdbc/.

[SUN03b] Sun Microsystems. Java Transaction API (JTA). 2003. http://java.sun.com/products/jta/.

[SUN03c] Sun Microsystems. The JNDI Tutorial. 2003. http://java.sun.com/products/jndi/tutorial/.

[SUN03d] Sun Microsystems. J2EE Connector Architecture. 2003. http://java.sun.com/j2ee/connector/.

[SUN03e] Sun Microsystems**.** Java API for XML Processing (JAXP) Documentation. 2003. http://java.sun.com/xml/jaxp/docs.html.

[SUN03f] Sun Microsystems**.** Java Remote Method Invocation (RMI). http://java.sun.com/products/jdk/rmi/.

[SUT95] Sutcliffe A., "Human-Computer Interface Design". 2nd ed. Houndsmills, Basingstoke, Hampshire: Macmillan Press Ltd, 1995.

[TIE02] Tierney B., Aydt R., Gunter D, Smith W., Taylor V., Wolski R., Swany M., "A Grid Monitoring Architecture" - The Global Grid Forum GWD-GP-16-2, January 2002.

[TUM04] Tummala K., "Glue Generation Framework In UniFrame for the CORBA-JAVA/RMI INTEROPERABILITY " M.S. Project TR-CIS-0302-03, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, May, 2004

[VNE03] User-Centred Product Creation in Interactive Electronic Publishing http://www.vnet5.org/

[WWW03] World Wide Web. HTTP - Hypertext Transfer Protocol. 2003. http://www.w3.org/Protocols/

APPENDICES

APPENDIX A: State Diagrams of URDS Entities

A state diagram can be defined as a diagram which “shows a sequence of states an object can assume during its lifetime, together with the stimuli that cause change of state” [OES99].

In this appendix the state diagrams for each entity of URDS is provided. Below is the description for understanding the state diagrams and followed by the acronyms used in the state diagrams:

- state changes takes place only when an event is fired; events are fired only if the condition in [ ] takes place (if exists)
- all the components are considered to run continuously, they reach a stop state only if there is an unrecoverable error (catastrophic error) or the system needs to be shutdown
- every state chart diagram has some associated error events, like type1 and type2 errors, on the occurrence of which a different remedial action takes place; depending upon the error encountered, an associated error message is displayed

Domain Security Manager State Diagram:

The following example shows the state transitions for a DSM. In this diagram each state is uniquely identified by a set of flags and description about each flag is presented along with its state diagram. The DSM is initialized and reaches the *Start* state, where its tries to connect to its repository. Upon successful connection it reaches the *DSM Ready* state and sets the *conn_rep_flag* to 1. If the connection is unsuccessful, it raises an event and reaches the *Error* state and sets the value of the *error_flag* to 1. Depending upon the type of error, an appropriate error message is displayed. After this action, the *error_flag* is reset and the control reaches the *Start* state, giving it an option to reconnect to the repository. When DSM is in *DSM Ready* state, it may receive requests from the principals or QM. Upon a request, the DSM reaches the *Process request* state and sets the *busy_flag* to 1. After the request has been processed, the control goes back to the *DSM Ready* state and resets the flag value to 0. If an error occurs in the *Process request* state, an event is raised. Subsequently, control goes to *Error* state where the *error_flag* is set to 1 and displays a suitable error message. After this action the *error_flag* is reset to 0 and control reaches *DSM Ready* state. The DSM reaches a *Stop* state if an unrecoverable error or a catastrophic failure occurs, in such a case *Stop* state is reached directly from any of the above mentioned states.

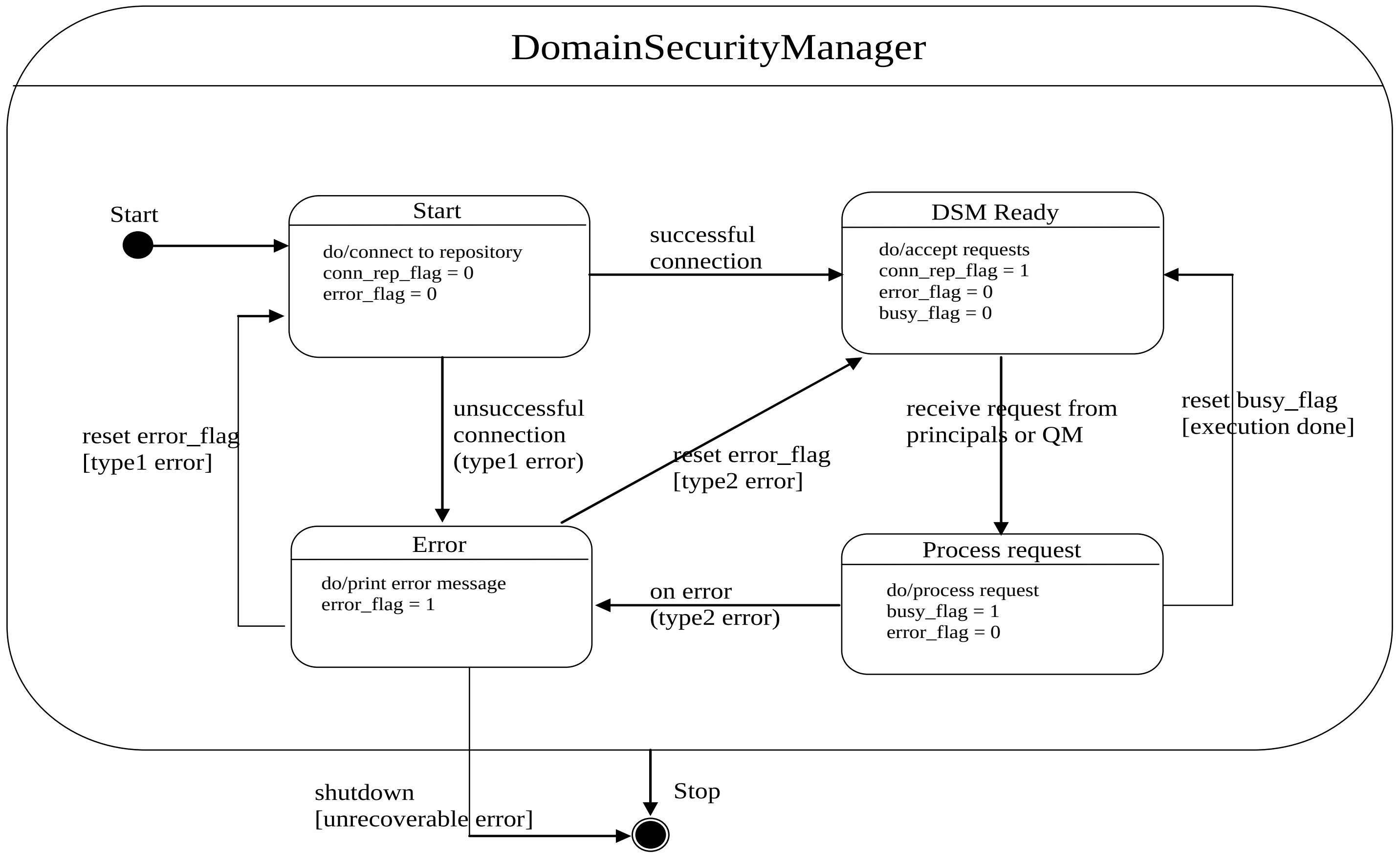


Figure a State Diagram of Domain Security Manager

All requests are considered as if they are of the same kind (requests from principals and QM)

- conn_rep_flag - flag to show the connection between DSM and its repository
- error_flag - this flag is set to 1 only if there is an error
- busy_flag - this flag is set to 1 if a request is being processed

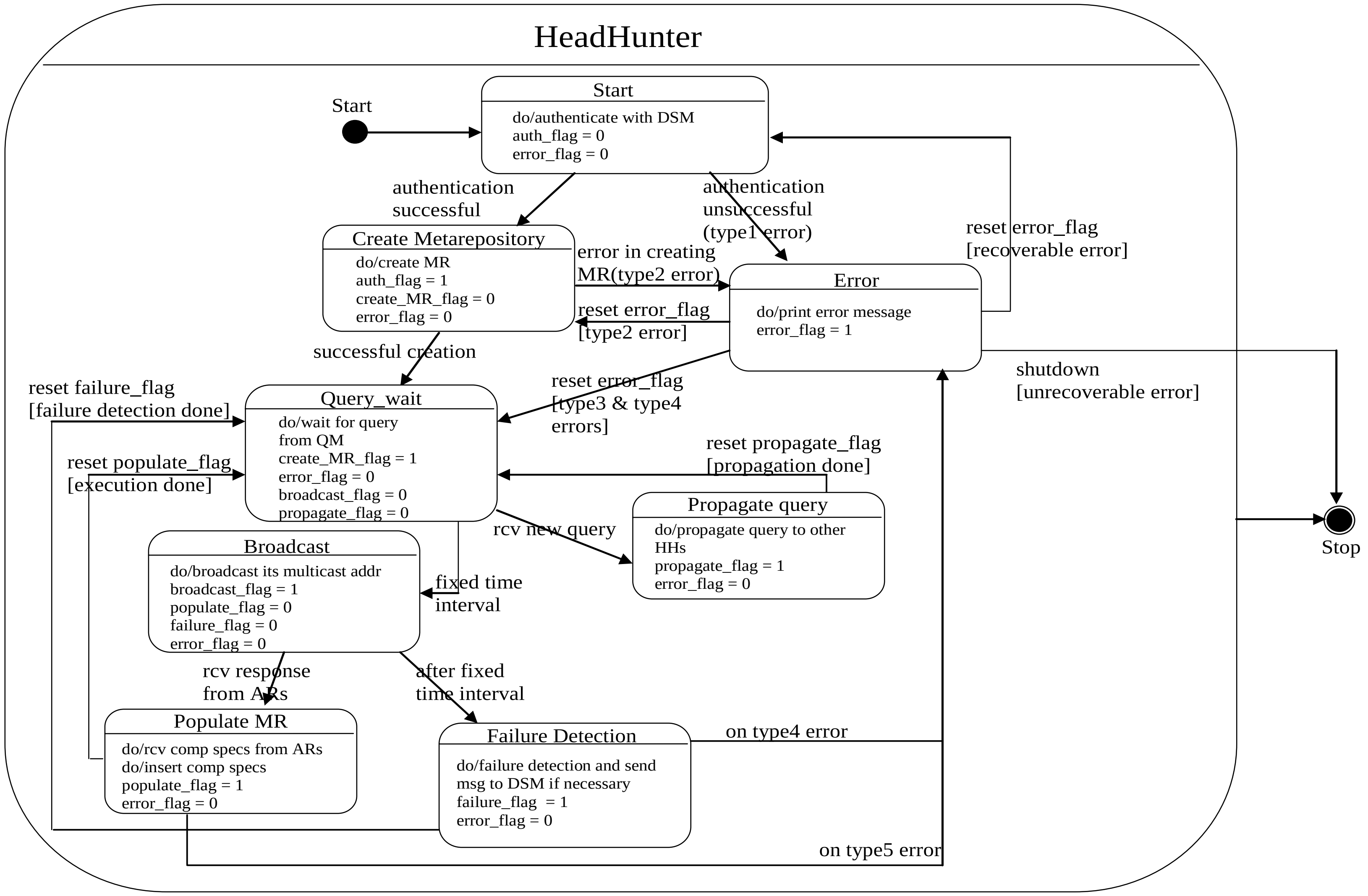


Figure b State Diagram of Headhunter

- auth_flag - flag to represent if AR is authenticated with DSM ;
- error_flag - this flag is set to 1 only if there is an error
- create_MR_flag - to represent if MR is created or not
- broadcast_flag - to represent if HH is broadcasting its multicast address or not
- popagate_flag - flag to represent if HH is propageting the query to other HHs
- populate_flag - this flag is set to 1 when HH receives comp specs from AR and stores them in MR
- failure_flag - this is set to 1 when the failure detection process is going on

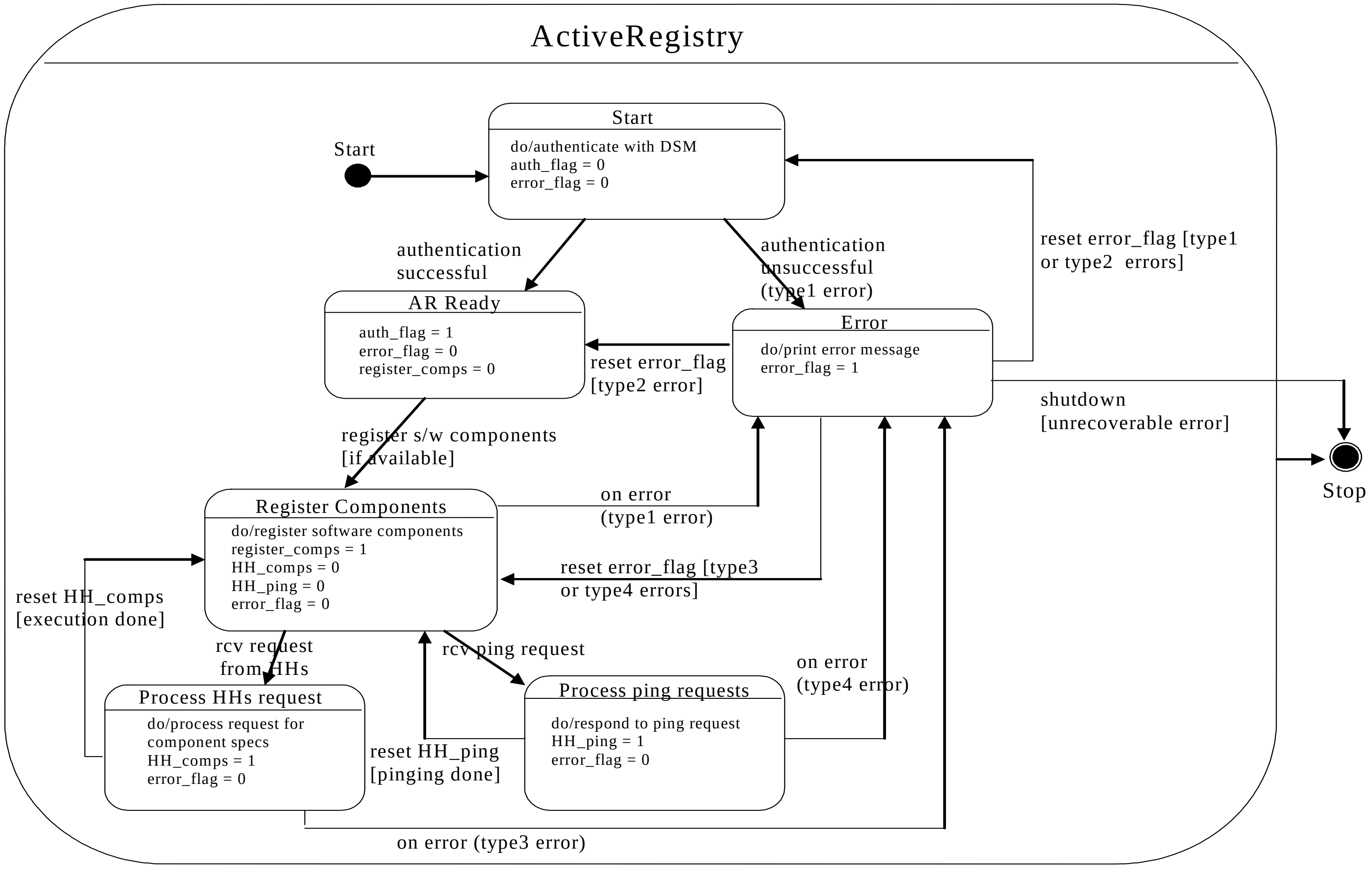


Figure c State Diagram of Active Registry

- auth_flag - flag to represent if AR is authenticated with DSM
- error_flag - this flag is set to 1 only if there is an error
- register_comps - flag to represent if AR is resgistring service components
- HH_comps - process HH’s request for component specifications
- HH_ping - process HH’s ping request

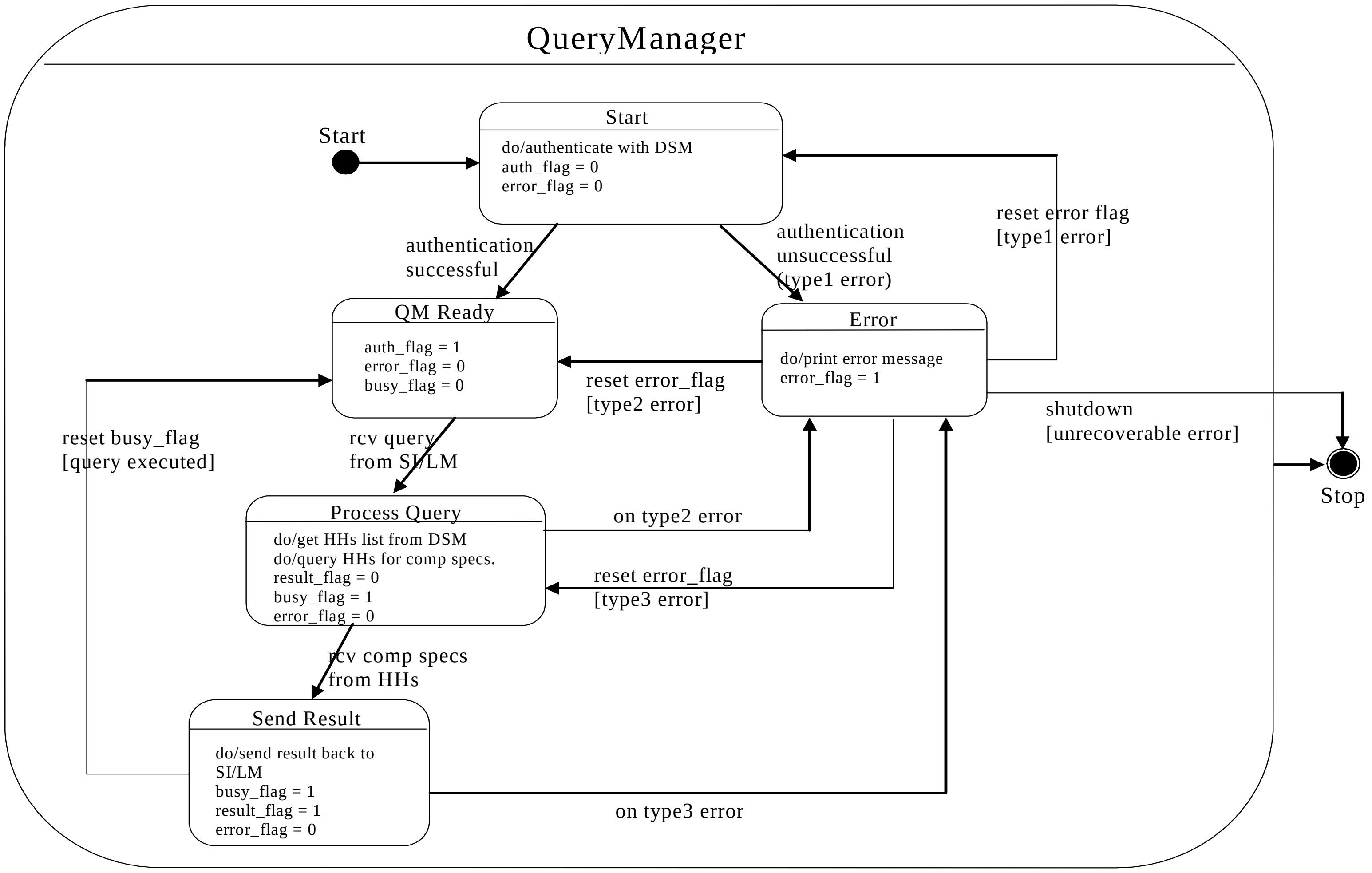


Figure d State Diagram of Query Manager

- auth_flag - flag to represent if AR is authenticated with DSM
- error_flag - this flag is set to 1 only if there is an error
- busy_flag - this flag is set to 1 if QM is processing a query
- result_flag - this flag is set to 1 if the resultant component specifications are available

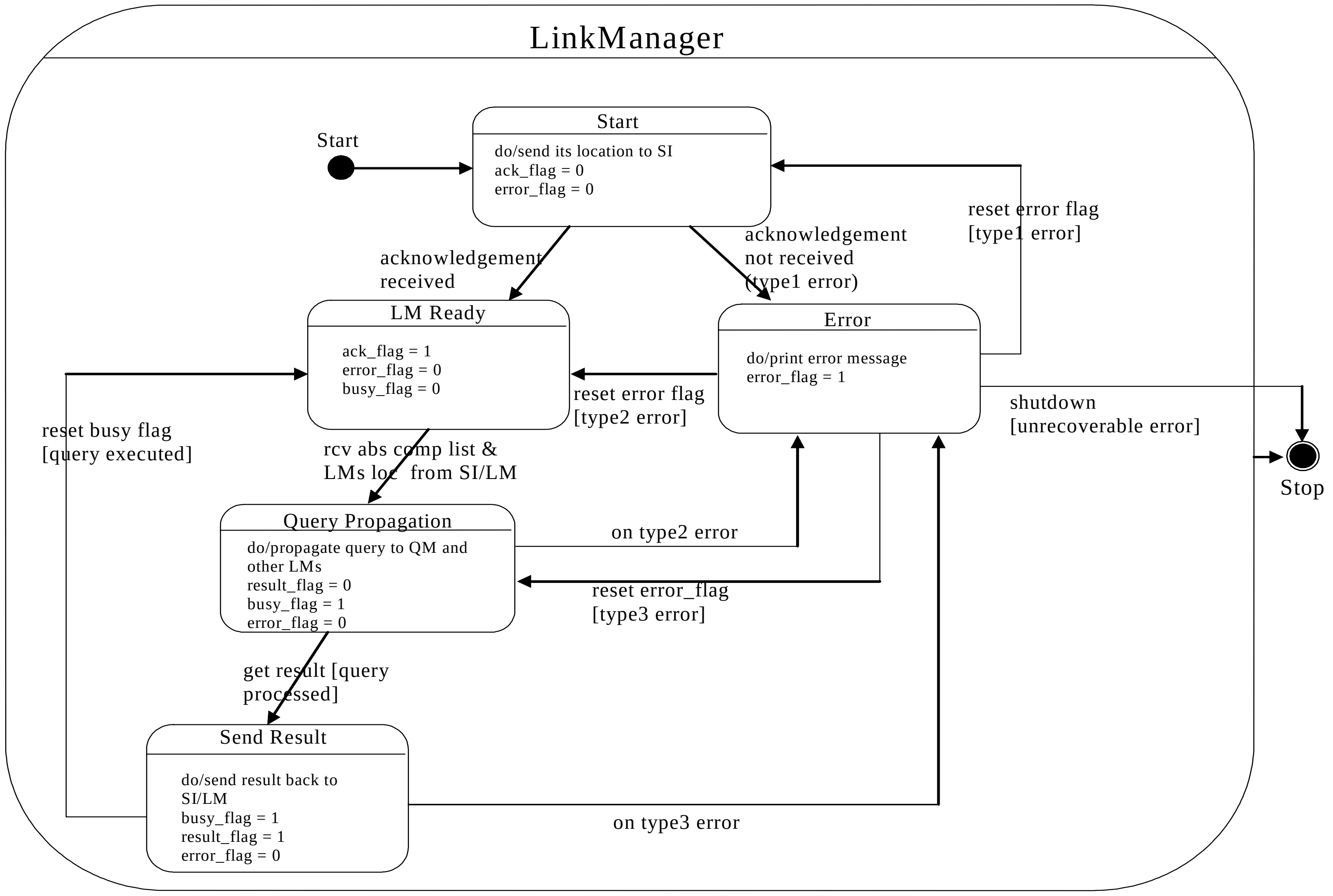

LinkManager
Start
Start
do/send its location to SI
ack_flag = 0
error_flag = 0
acknowledgement received
acknowledgement not received (type1 error)
reset error flag [type1 error]
LM Ready
ack_flag = 1
error_flag = 0
busy_flag = 0
Error
do/print error message
error_flag = 1
reset error flag [type2 error]
shutdown [unrecoverable error]
Stop
reset busy flag [query executed]
rcv abs comp list & LMs loc from SI/LM
Query Propagation
do/propagate query to QM and other LMs
result_flag = 0
busy_flag = 1
error_flag = 0
on type2 error
reset error_flag [type3 error]
get result [query processed]
Send Result
do/send result back to SI/LM
busy_flag = 1
result_flag = 1
error_flag = 0
on type3 error

Figure e State Diagram of Link Manager

- ack_flag - represents if it receives an acknowledgement from SG
- error_flag - this flag is set to 1 only if there is an error
- busy_flag - this flag is set to 1 if LM is propagating a query to QM or other LM’s
- result_flag - this flag is set to 1 if the resultant component specifications are available

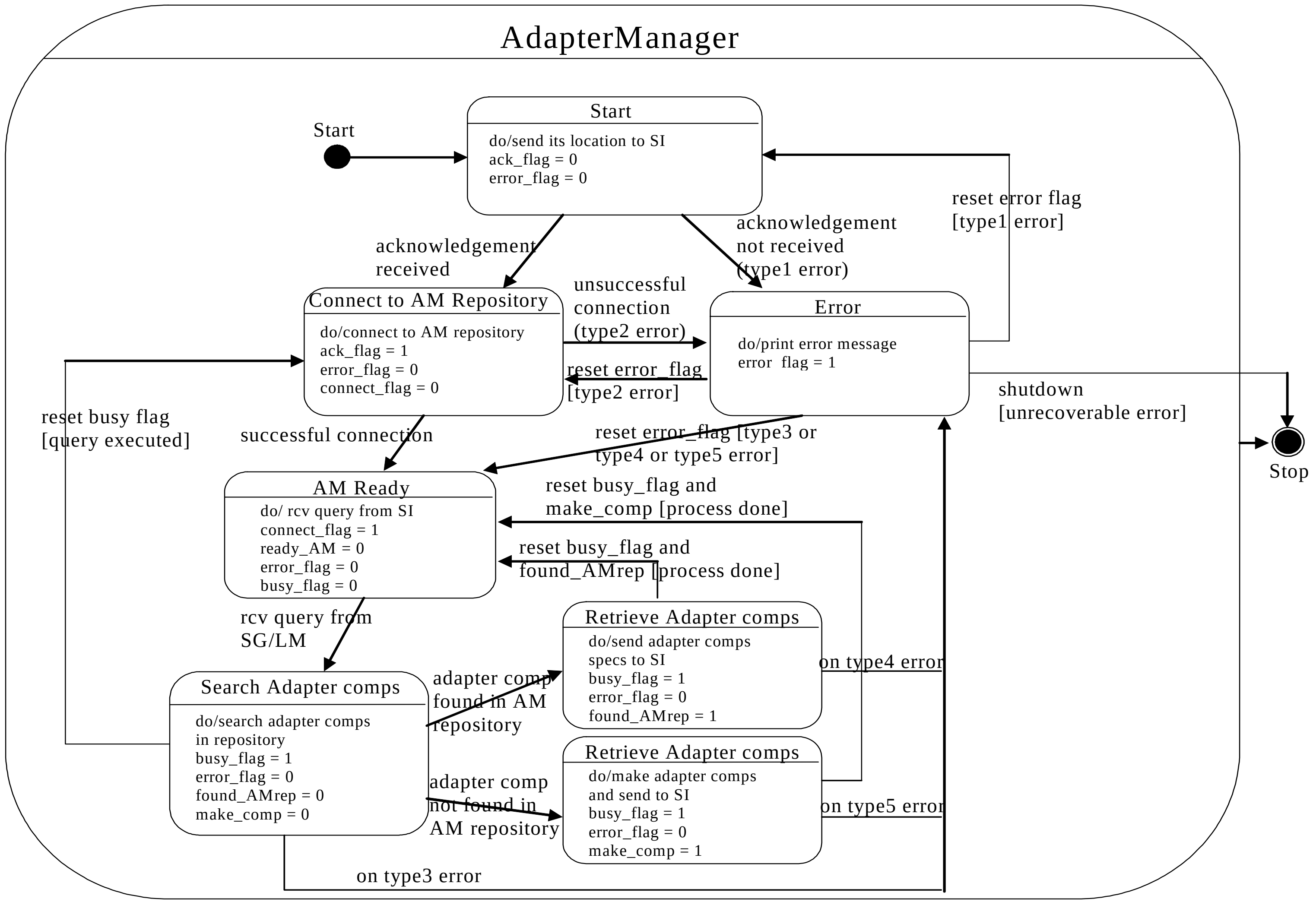
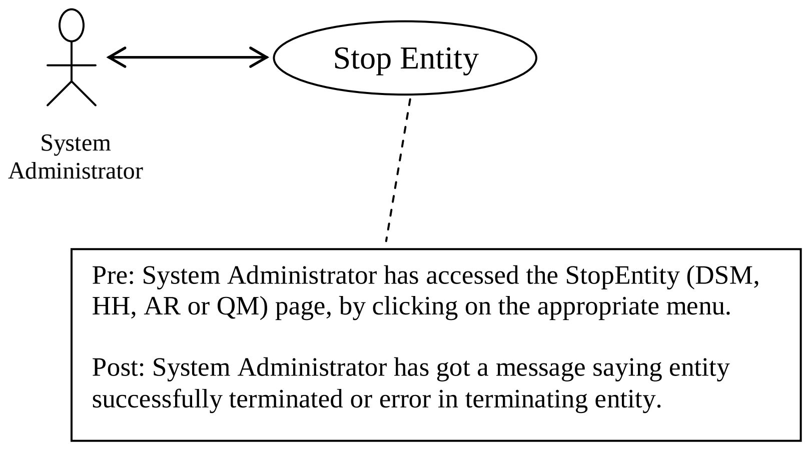

AdapterManager
Start
Start
do/send its location to SI
ack_flag = 0
error_flag = 0
acknowledgement received
acknowledgement not received (type1 error)
reset error flag [type1 error]
Connect to AM Repository
do/connect to AM repository
ack_flag = 1
error_flag = 0
connect_flag = 0
unsuccessful connection (type2 error)
reset error_flag [type2 error]
Error
do/print error message
error flag = 1
shutdown [unrecoverable error]
Stop
reset busy flag [query executed]
successful connection
reset error_flag [type3 or type4 or type5 error]
AM Ready
do/ rcv query from SI
connect_flag = 1
ready_AM = 0
error_flag = 0
busy_flag = 0
reset busy_flag and make_comp [process done]
reset busy_flag and found_AMrep [process done]
rcv query from SG/LM
Retrieve Adapter comps
do/send adapter comps specs to SI
busy_flag = 1
error_flag = 0
found_AMrep = 1
on type4 error
Search Adapter comps
do/search adapter comps in repository
busy_flag = 1
error_flag = 0
found_AMrep = 0
make_comp = 0
adapter comp found in AM repository
adapter comp not found in AM repository
Retrieve Adapter comps
do/make adapter comps and send to SI
busy_flag = 1
error_flag = 0
make_comp = 1
on type5 error
on type3 error

Figure f State Diagram of Adapter Manager

- ack_flag - flag to represent if AM receives an acknowledgement from SG or not
- error_flag - this flag is set to 1 only if there is an error
- busy_flag - this flag is set to 1 if AM is processing a query from SG
- connect_flag - this flag is set to 1 if a connection is established between AM and its repository
- found_AMrep - this flag is set if AM finds an already existing adapter component in its repository
- make_comp - this flag is set if AM need to synthesize adapter components

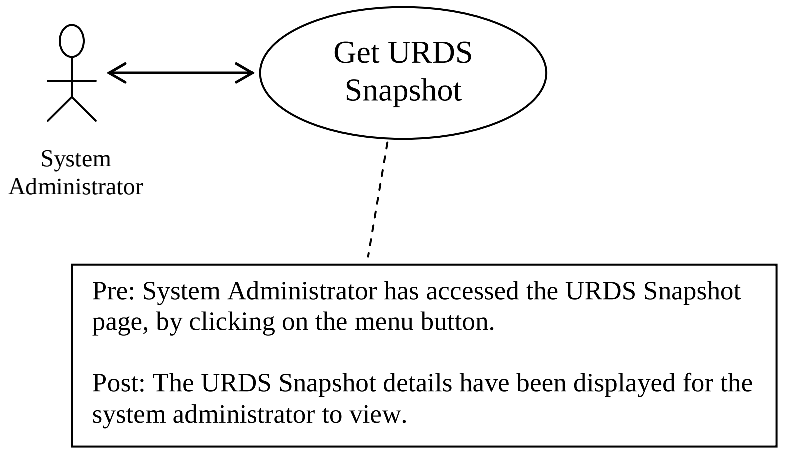
APPENDIX B: Use Case Diagrams



Pre: System Administrator has accessed the StopEntity (DSM, HH, AR or QM) page, by clicking on the appropriate menu.

Post: System Administrator has got a message saying entity successfully terminated or error in terminating entity.

Figure i Use Case Diagram for terminating an entity



Pre: System Administrator has accessed the URDS Snapshot page, by clicking on the menu button.

Post: The URDS Snapshot details have been displayed for the system administrator to view.

Figure ii Use Case Diagram for retrieving URDS Snapshot

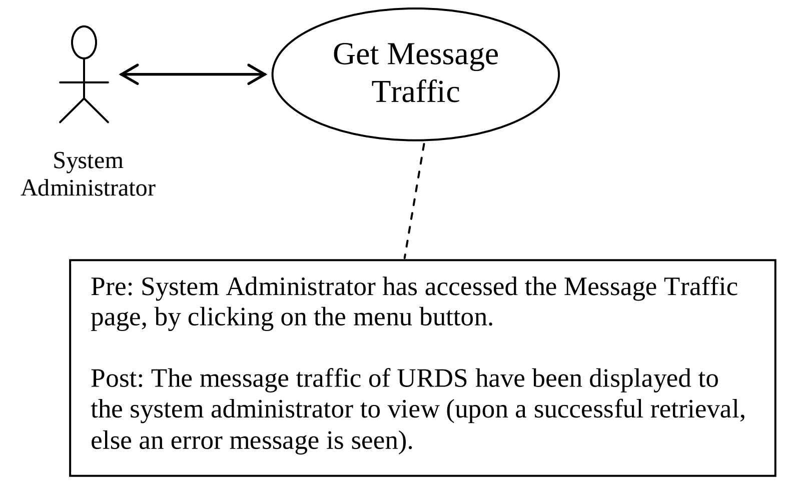

Get Message Traffic
System Administrator
Pre: System Administrator has accessed the Message Traffic page, by clicking on the menu button.
Post: The message traffic of URDS have been displayed to the system administrator to view (upon a successful retrieval, else an error message is seen).

Figure iii Use Case Diagram for retrieving Message Traffic

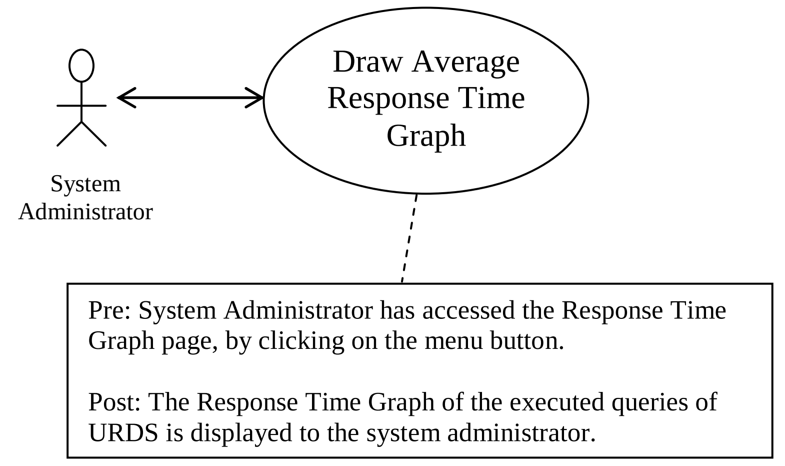

Draw Average Response Time Graph
System Administrator
Pre: System Administrator has accessed the Response Time Graph page, by clicking on the menu button.
Post: The Response Time Graph of the executed queries of URDS is displayed to the system administrator.

Figure iv Use Case Diagram for display of average response time graph

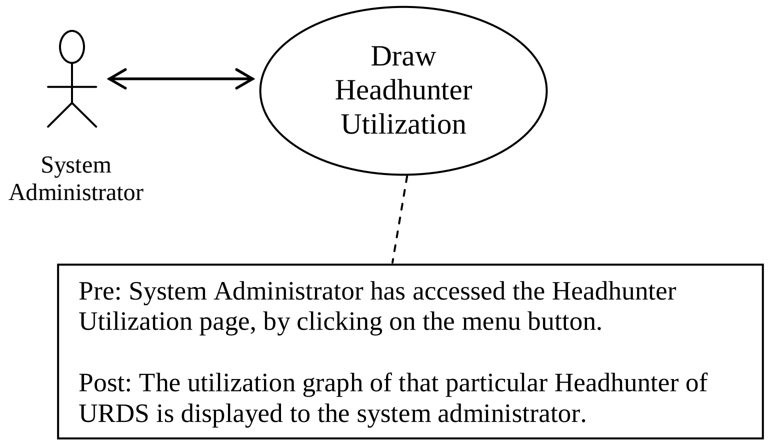

System Administrator
Draw Headhunter Utilization
Pre: System Administrator has accessed the Headhunter Utilization page, by clicking on the menu button.
Post: The utilization graph of that particular Headhunter of URDS is displayed to the system administrator.

Figure v Use Case Diagram for display of Headhunter utilization graph

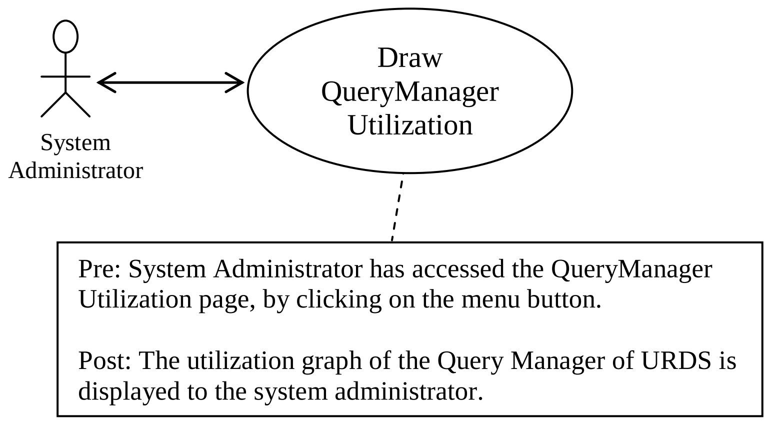

System Administrator
Draw QueryManager Utilization
Pre: System Administrator has accessed the QueryManager Utilization page, by clicking on the menu button.
Post: The utilization graph of the Query Manager of URDS is displayed to the system administrator.

Figure vi Use Case Diagram for display of Query Manager Utilization graph

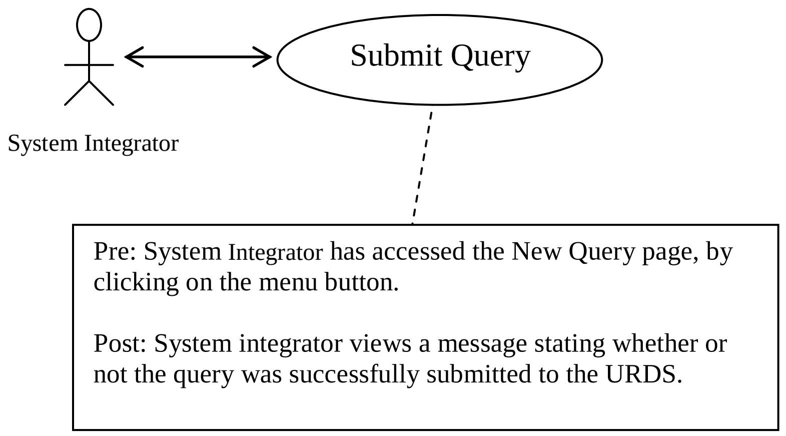

Submit Query
System Integrator
Pre: System Integrator has accessed the New Query page, by clicking on the menu button.
Post: System integrator views a message stating whether or not the query was successfully submitted to the URDS.

Figure vii Use Case Diagram for submitting a query to URDS

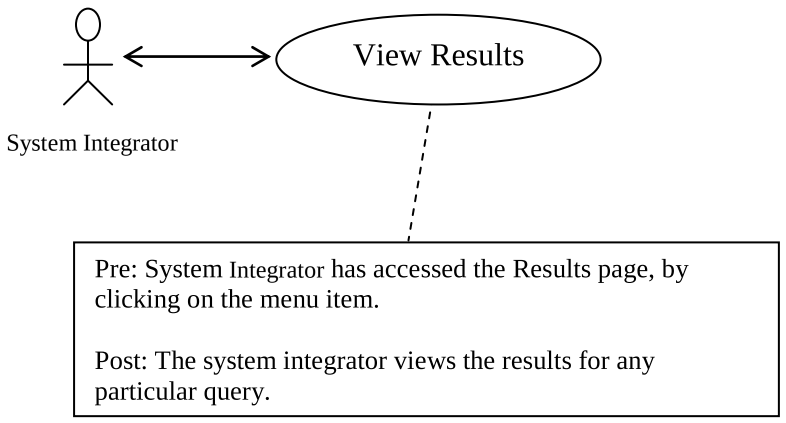
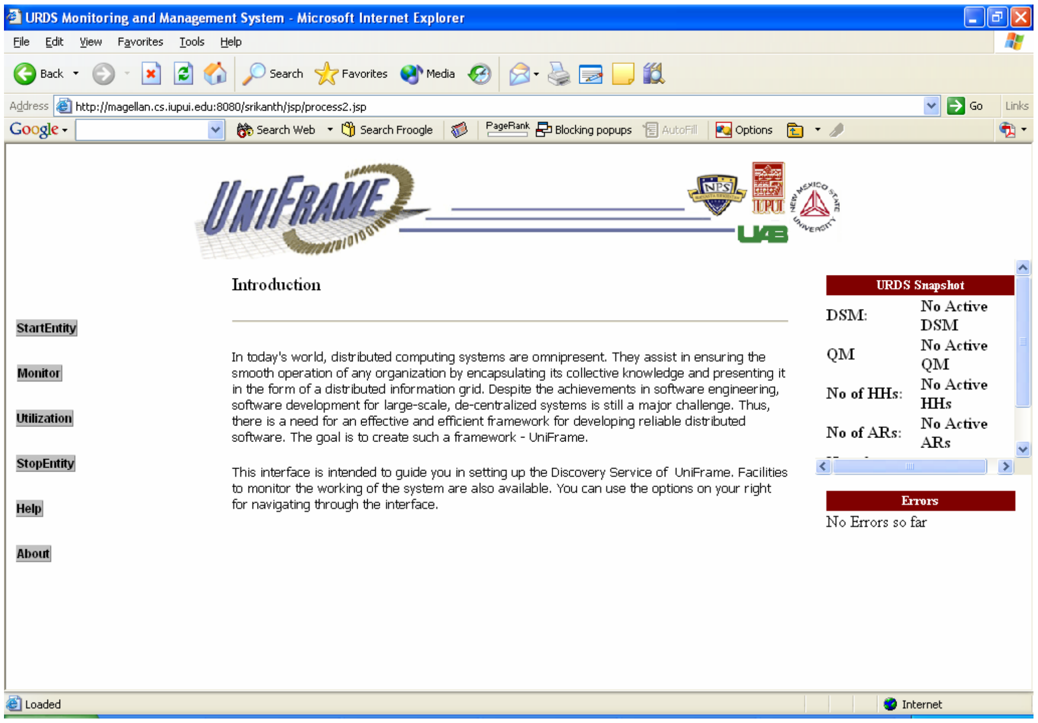

View Results
System Integrator
Pre: System Integrator has accessed the Results page, by clicking on the menu item.
Post: The system integrator views the results for any particular query.

Figure viii Use Case Diagram for viewing results of a query

APPENDIX C: Screen Shots of URDSMMS


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

Introduction

In today's world, distributed computing systems are omnipresent. They assist in ensuring the smooth operation of any organization by encapsulating its collective knowledge and presenting it in the form of a distributed information grid. Despite the achievements in software engineering, software development for large-scale, de-centralized systems is still a major challenge. Thus, there is a need for an effective and efficient framework for developing reliable distributed software. The goal is to create such a framework - UniFrame.

This interface is intended to guide you in setting up the Discovery Service of UniFrame. Facilities to monitor the working of the system are also available. You can use the options on your right for navigating through the interface.

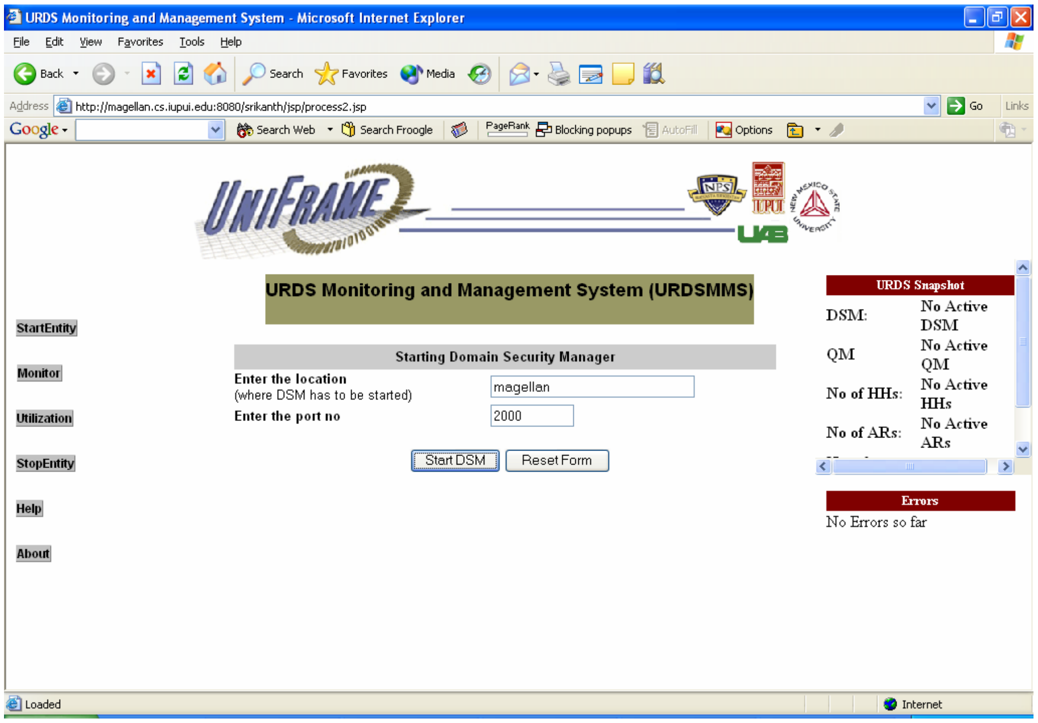
URDS Snapshot

DSM:	No Active DSM
QM	No Active QM
No of HHs:	No Active HHs
No of ARs:	No Active ARs

Errors

No Errors so far

Figure i: View showing the main page of system administrator view


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Start DSM
Reset Form
Loaded
Internet

URDS Monitoring and Management System (URDSMMS)

Starting Domain Security Manager

Enter the location (where DSM has to be started)	magellan
Enter the port no	2000

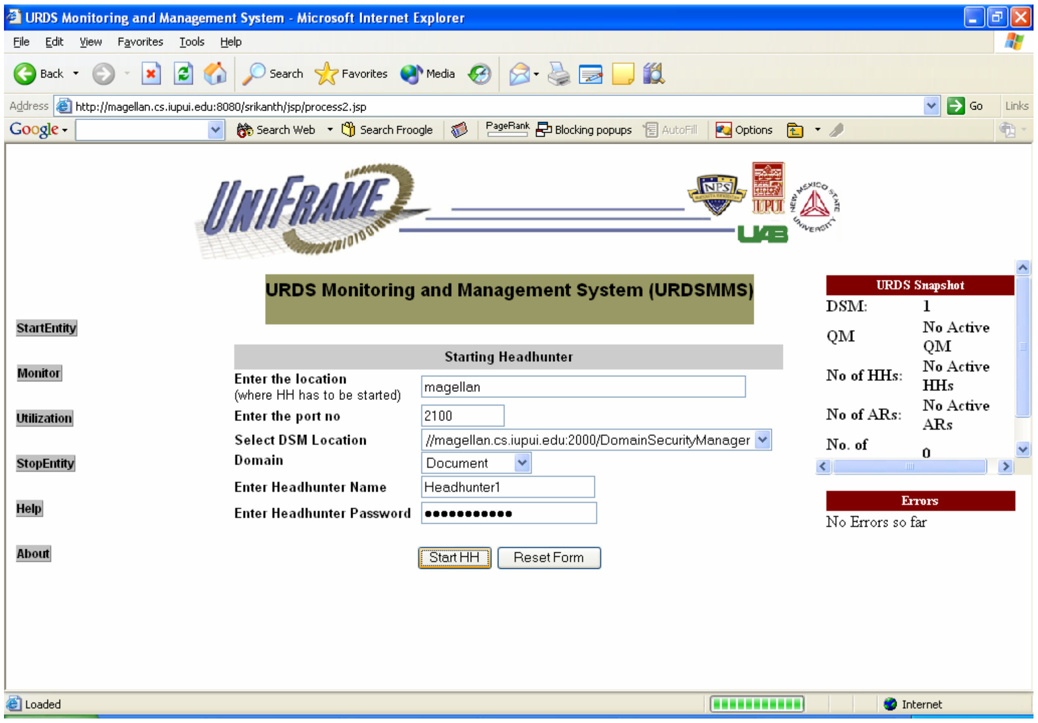
URDS Snapshot

DSM:	No Active DSM
QM	No Active QM
No of HHs:	No Active HHs
No of ARs:	No Active ARs

Errors

No Errors so far

Figure ii: View showing the initiation of a Domain Security Manager


URDS Monitoring and Management System - Microsoft Internet Explorer

http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp

URDS Monitoring and Management System (URDSMMS)

StartEntity
Monitor
Utilization
StopEntity
Help
About

Starting Headhunter

Enter the location (where HH has to be started)	magellan
Enter the port no	2100
Select DSM Location	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Domain	Document
Enter Headhunter Name	Headhunter1
Enter Headhunter Password	●●●●●●●●●●●

Start HH | Reset Form

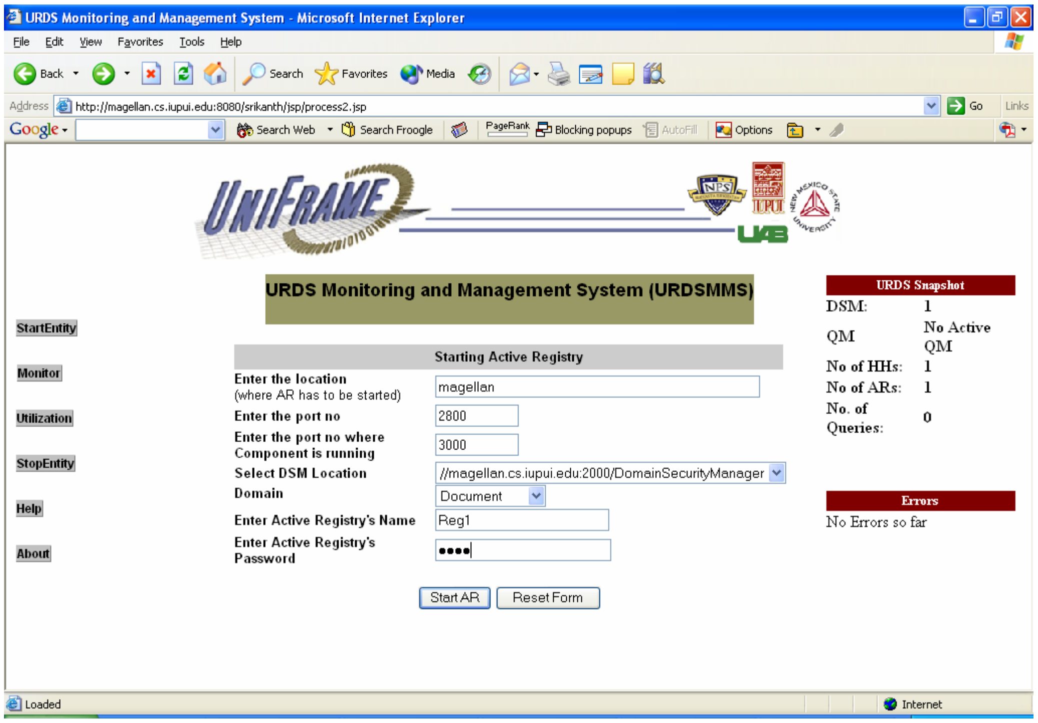
URDS Snapshot

DSM:	1
QM	No Active QM
No of HHs:	No Active HHs
No of ARs:	No Active ARs
No. of	0

Errors

No Errors so far

Figure iii: View showing the initiation of a Headhunter


URDS Monitoring and Management System - Microsoft Internet Explorer

http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp

URDS Monitoring and Management System (URDSMMS)

StartEntity
Monitor
Utilization
StopEntity
Help
About

Starting Active Registry

Enter the location (where AR has to be started)	magellan
Enter the port no	2800
Enter the port no where Component is running	3000
Select DSM Location	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Domain	Document
Enter Active Registry's Name	Reg1
Enter Active Registry's Password	●●●●

Start AR | Reset Form

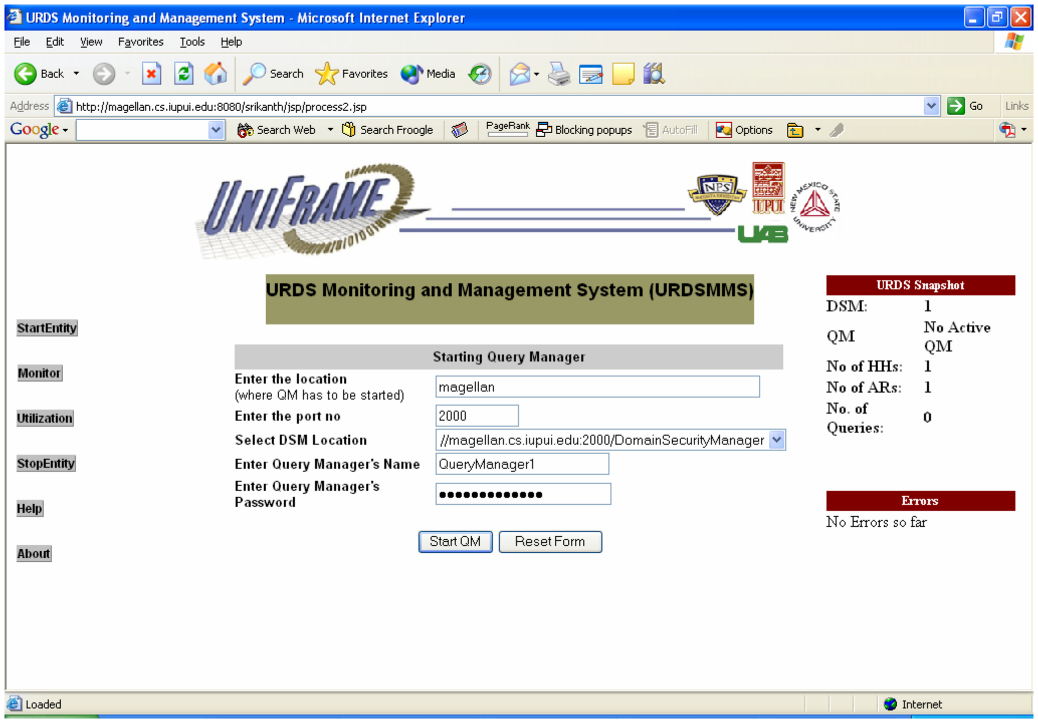
URDS Snapshot

DSM:	1
QM	No Active QM
No of HHs:	1
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure iv: View showing the initiation of an Active Registry


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
Search Web Search Froogle PageRank Blocking popups AutoFill Options
UNIFRAME
NPS IUPUI UAB
StartEntity
Monitor
Utilization
StopEntity
Help
About
Start QM
Reset Form
Loaded
Internet

URDS Monitoring and Management System (URDSMMS)

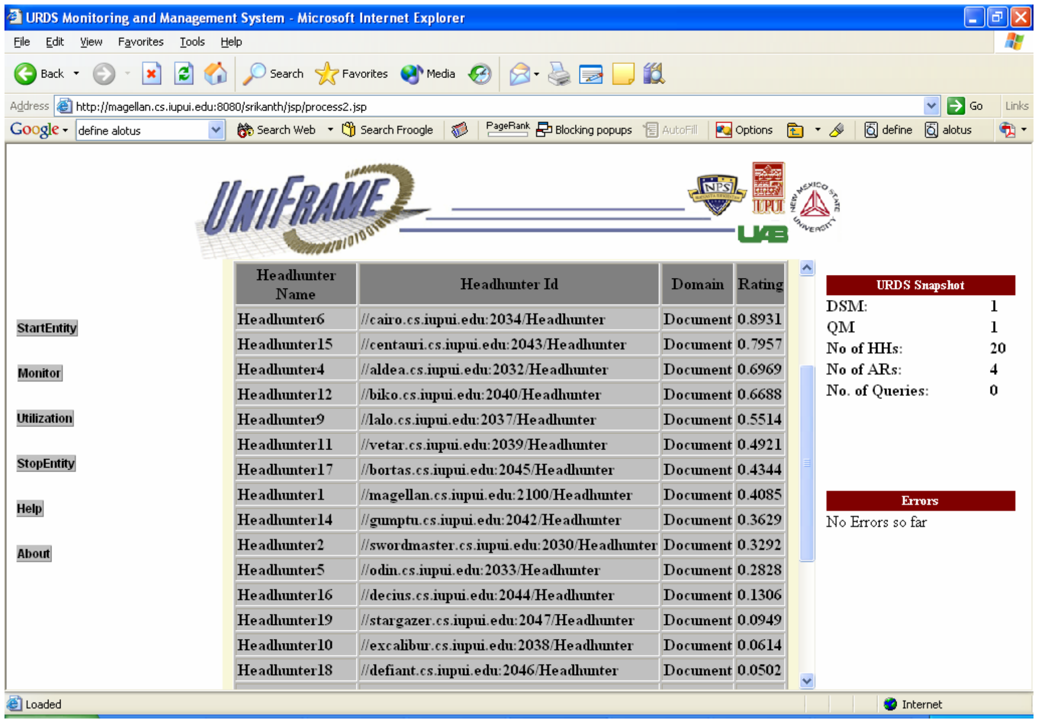
Starting Query Manager	
Enter the location (where QM has to be started)	magellan
Enter the port no	2000
Select DSM Location	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Enter Query Manager's Name	QueryManager1
Enter Query Manager's Password	●●●●●●●●●●●●●

URDS Snapshot	
DSM:	1
QM	No Active QM
No of HHs:	1
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure v: View showing the initiation of a Query Manager


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
define alotus
Search Web Search Froogle PageRank Blocking popups AutoFill Options define alotus
UNIFRAME
NPS IUPUI UAB
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

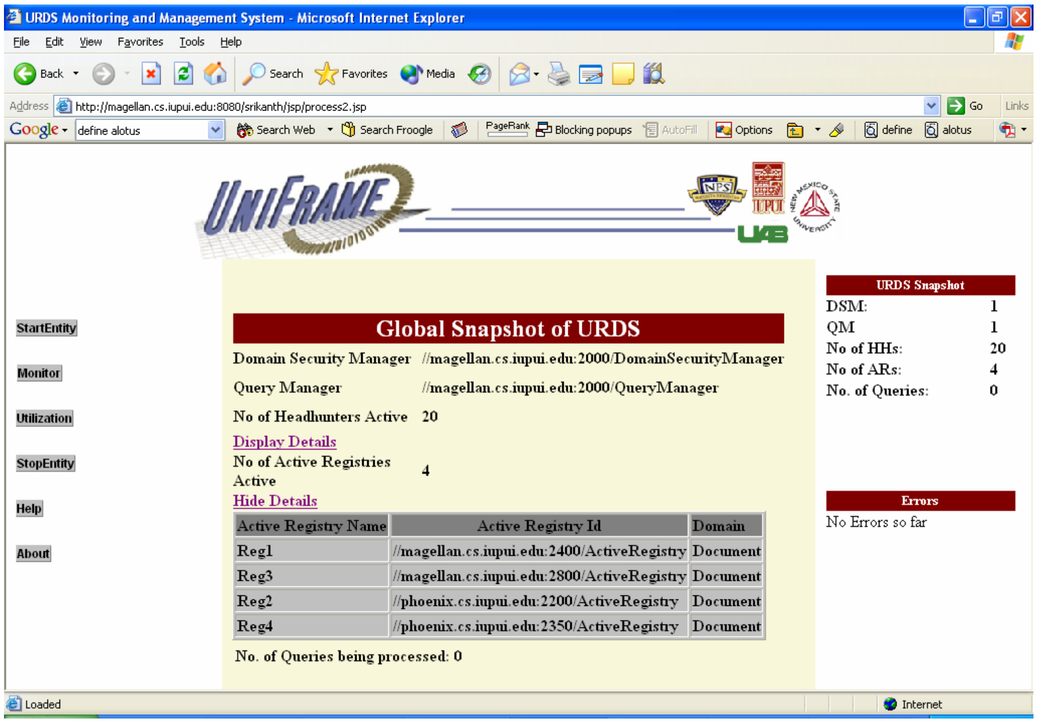
Headhunter Name	Headhunter Id	Domain	Rating
Headhunter6	//cairo.cs.iupui.edu:2034/Headhunter	Document	0.8931
Headhunter15	//centauri.cs.iupui.edu:2043/Headhunter	Document	0.7957
Headhunter4	//aldea.cs.iupui.edu:2032/Headhunter	Document	0.6969
Headhunter12	//biko.cs.iupui.edu:2040/Headhunter	Document	0.6688
Headhunter9	//lalo.cs.iupui.edu:2037/Headhunter	Document	0.5514
Headhunter11	//vetar.cs.iupui.edu:2039/Headhunter	Document	0.4921
Headhunter17	//bortas.cs.iupui.edu:2045/Headhunter	Document	0.4344
Headhunter1	//magellan.cs.iupui.edu:2100/Headhunter	Document	0.4085
Headhunter14	//gumptu.cs.iupui.edu:2042/Headhunter	Document	0.3629
Headhunter2	//swordmaster.cs.iupui.edu:2030/Headhunter	Document	0.3292
Headhunter5	//odin.cs.iupui.edu:2033/Headhunter	Document	0.2828
Headhunter16	//decius.cs.iupui.edu:2044/Headhunter	Document	0.1306
Headhunter19	//stargazer.cs.iupui.edu:2047/Headhunter	Document	0.0949
Headhunter10	//excalibur.cs.iupui.edu:2038/Headhunter	Document	0.0614
Headhunter18	//defiant.cs.iupui.edu:2046/Headhunter	Document	0.0502

URDS Snapshot	
DSM:	1
QM	1
No of HHs:	20
No of ARs:	4
No. of Queries:	0

Errors

No Errors so far

Figure vi: View showing the details of Headhunters



URDS Monitoring and Management System - Microsoft Internet Explorer

Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp

Global Snapshot of URDS

Domain Security Manager	//magellan.cs.iupui.edu:2000/DomainSecurityManager
Query Manager	//magellan.cs.iupui.edu:2000/QueryManager
No of Headhunters Active	20

Display Details

No of Active Registries Active	4

Hide Details

Active Registry Name	Active Registry Id	Domain
Reg1	//magellan.cs.iupui.edu:2400/ActiveRegistry	Document
Reg3	//magellan.cs.iupui.edu:2800/ActiveRegistry	Document
Reg2	//phoenix.cs.iupui.edu:2200/ActiveRegistry	Document
Reg4	//phoenix.cs.iupui.edu:2350/ActiveRegistry	Document

No. of Queries being processed: 0

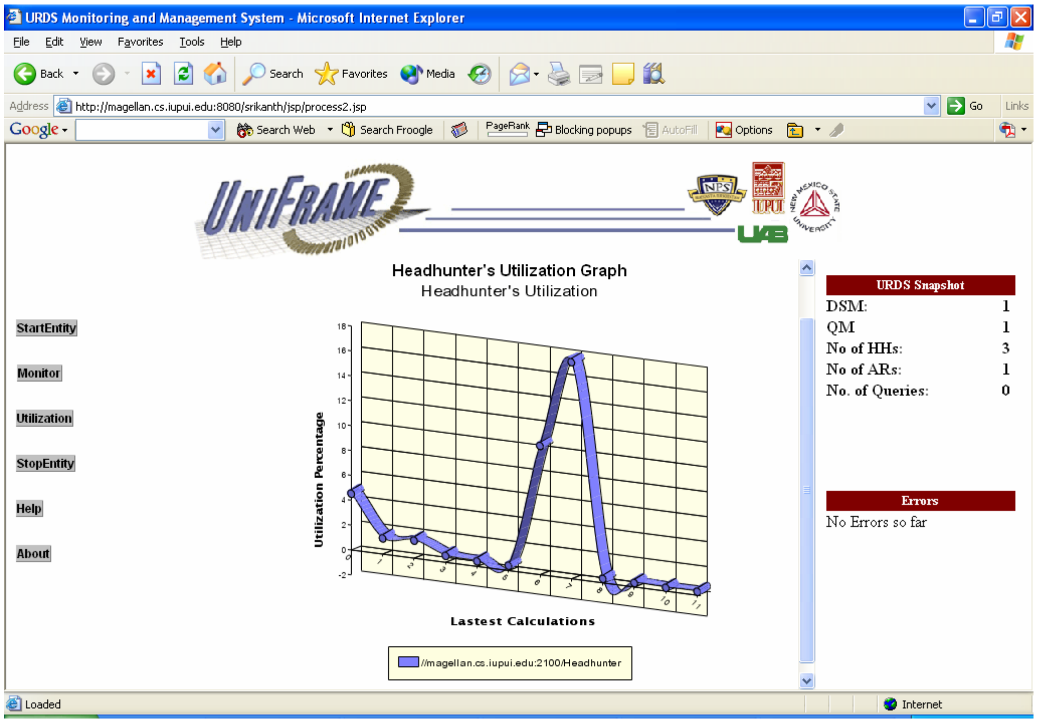
URDS Snapshot

DSM:	1
QM	1
No of HHs:	20
No of ARs:	4
No. of Queries:	0

Errors

No Errors so far

Figure vii: View showing the details of Active Registries



URDS Monitoring and Management System - Microsoft Internet Explorer

Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp

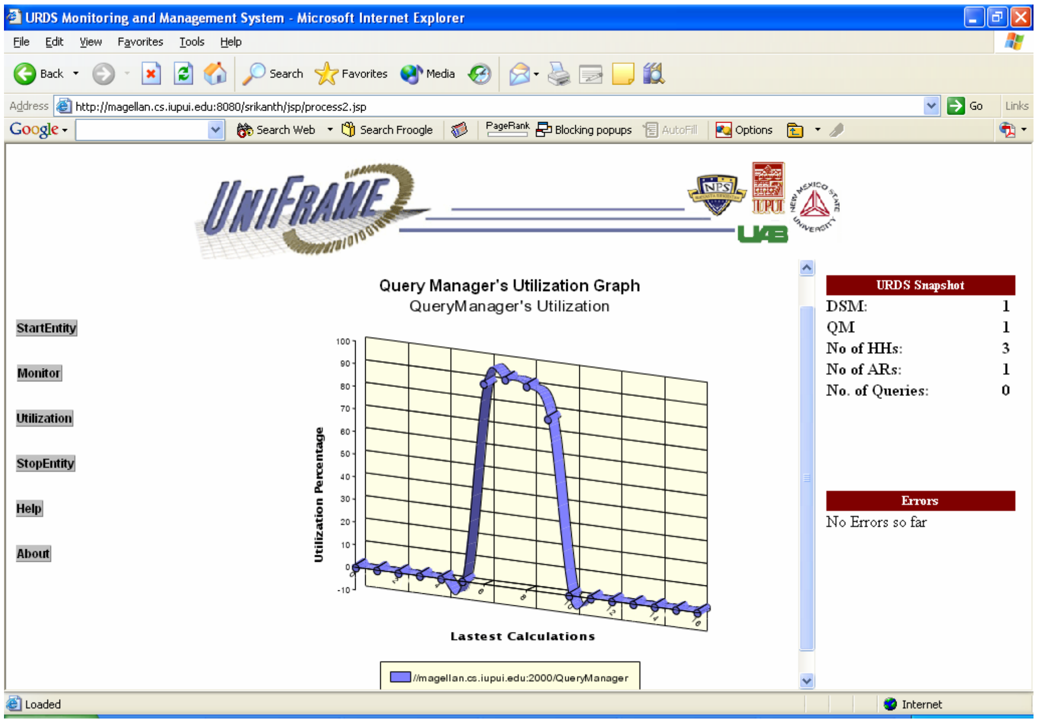
URDS Snapshot

DSM:	1
QM	1
No of HHs:	3
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure viii: View showing the graph of Headhunter’s utilization

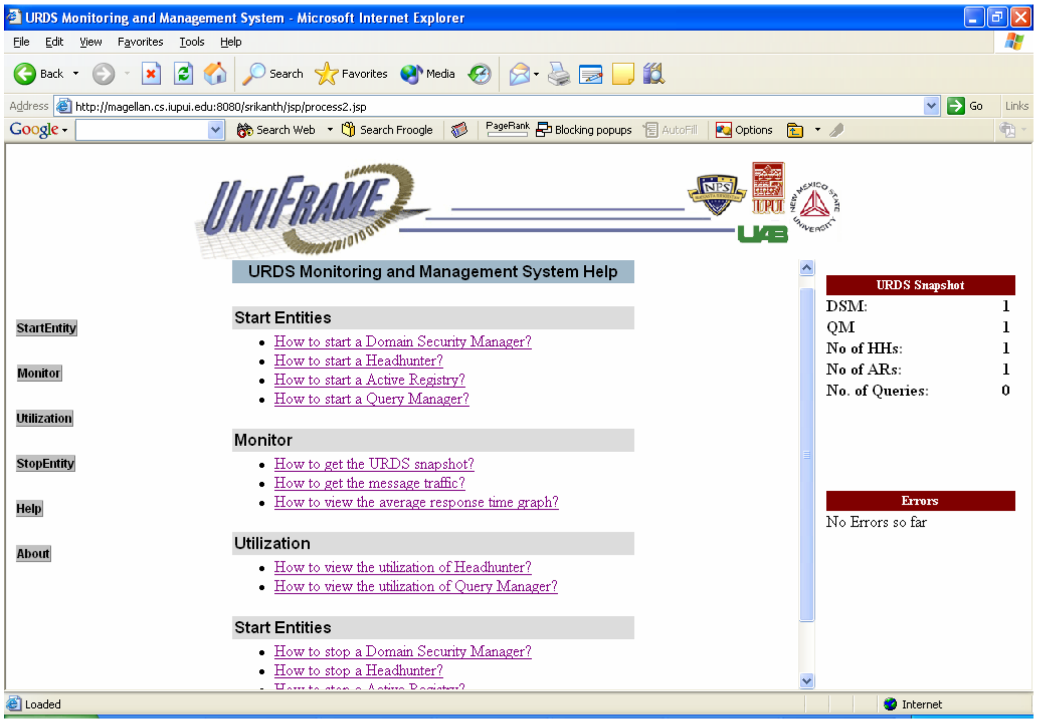

URDS Monitoring and Management System - Microsoft Internet Explorer
http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Query Manager's Utilization Graph
QueryManager's Utilization
Utilization Percentage
Lastest Calculations
//magellan.cs.iupui.edu:2000/QueryManager
Loaded
Internet

URDS Snapshot	
DSM:	1
QM	1
No of HHs:	3
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure ix: View showing the graph of Query Manager's utilization


URDS Monitoring and Management System - Microsoft Internet Explorer
http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Loaded
Internet

URDS Monitoring and Management System Help

Start Entities
- How to start a Domain Security Manager?
- How to start a Headhunter?
- How to start a Active Registry?
- How to start a Query Manager?

Monitor
- How to get the URDS snapshot?
- How to get the message traffic?
- How to view the average response time graph?

Utilization
- How to view the utilization of Headhunter?
- How to view the utilization of Query Manager?

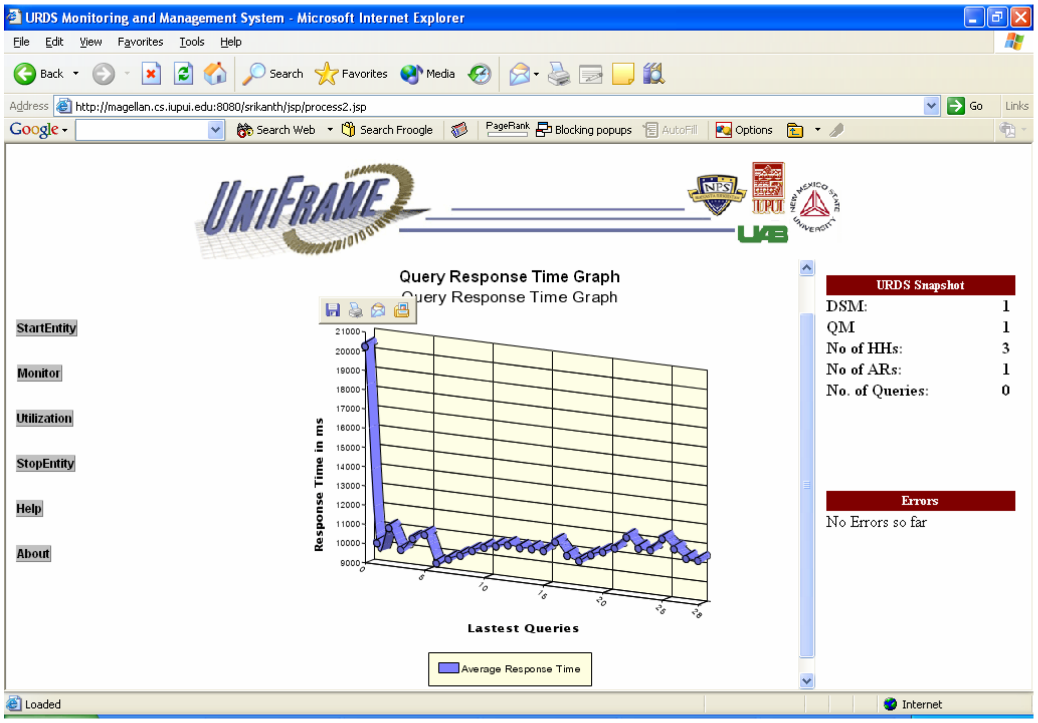
Start Entities
- How to stop a Domain Security Manager?
- How to stop a Headhunter?

URDS Snapshot	
DSM:	1
QM	1
No of HHs:	1
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure x: View showing the system help of URDSMMS (system administrator view)

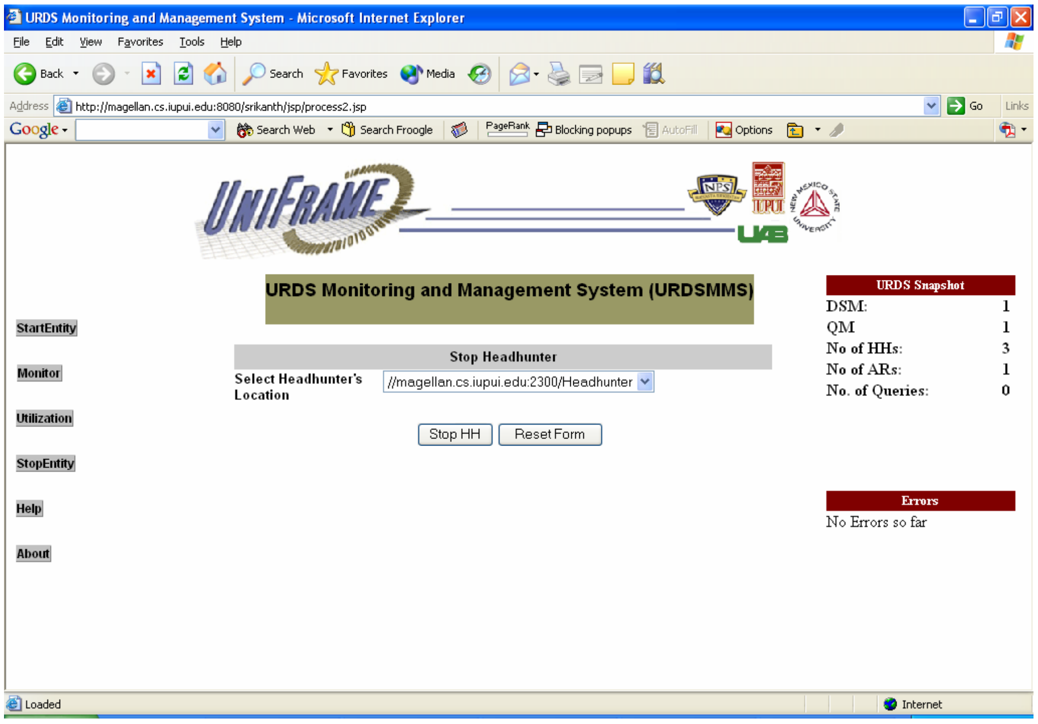

URDS Monitoring and Management System - Microsoft Internet Explorer
http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Query Response Time Graph
Response Time in ms
Lastest Queries
Average Response Time

URDS Snapshot	
DSM:	1
QM	1
No of HHs:	3
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure xi: View showing the graph of average query response time


URDS Monitoring and Management System - Microsoft Internet Explorer
http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
StartEntity
Monitor
Utilization
StopEntity
Help
About
Stop HH
Reset Form

URDS Monitoring and Management System (URDSMMS)

Stop Headhunter

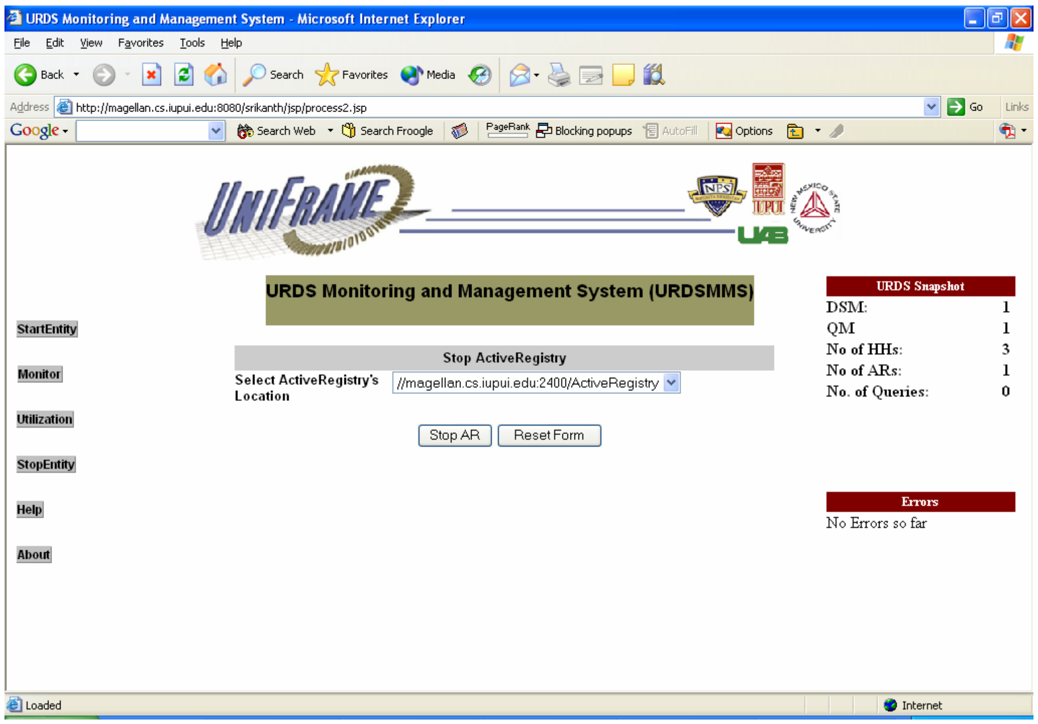
Select Headhunter's Location: //magellan.cs.iupui.edu:2300/Headhunter

URDS Snapshot	
DSM:	1
QM	1
No of HHs:	3
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure xii: View showing the termination of a Headhunter


URDS Monitoring and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://magellan.cs.iupui.edu:8080/srikanth/jsp/process2.jsp
Go Links
Search Web Search Froogle PageRank Blocking popups AutoFill Options
StartEntity
Monitor
Utilization
StopEntity
Help
About
Stop AR
Reset Form
Loaded
Internet

URDS Monitoring and Management System (URDSMMS)

Stop ActiveRegistry

Select ActiveRegistry's Location: //magellan.cs.iupui.edu:2400/ActiveRegistry

URDS Snapshot

DSM:	1
QM	1
No of HHs:	3
No of ARs:	1
No. of Queries:	0

Errors

No Errors so far

Figure xiii: View showing the termination of a Active Registry

APPENDIX D: User Surveys

1. *URDSMMS User Survey*

This appendix provides the actual user survey presented to UniFrame team members as noted in chapter 4. It consists of two separate types of questions, 1) UniFrame Specific that relate to questions on UniFrame and features of URDS, and 2) User Interface Specific which relates to question of the type of menu system.

URDS Monitoring and Management System (URDSMMS) User Survey

UniFrame Specific

- How would you rate your familiarity with the UniFrame project?
  (poor) 1 2 3 4 5 (great)

- How would you rate your familiarity with the UniFrame Resource Discovery Service (URDS)?
  (poor) 1 2 3 4 5 (great)

- How important is it to monitor the URDS?
  (little) 1 2 3 4 5 (very imp)

- How important is managing the URDS via an easy-to-use interface?
  (little) 1 2 3 4 5 (very imp)

- Do you think that the URDSMMS should contain two views, namely administrator view and system integrator/user view? Do you suggest any other view, if so please mention.
  (yes/no)

- Rate the features by their importance, that you would like to see in URDSMMS, on a scale of 1 (least important) to 5 (very important)
  a. Remote or local initiation and killing of different entities of URDS (such as Headhunters, Active Registries, Domain Security Manager and Query Manager) _____
  b. Submitting a query for searching components _____
  c. Representing the response time of a query in a graphical format with respect to time _____
  d. Showing the statistical and individual details of all active entities of URDS in a single snap-shot _____
  e. Checking if the Headhunters are active or not _____
  f. Providing help for the URDSMMS _____
  g. Displaying the utilization of Headhunter or Query Manager at any point of time

(so that we can know how much time the Headhunter or Query manager is busy) _____

User Interface Specific

- Do you feel comfortable with a menu driven (menus and sub-menus) interface? (Yes/No, if no please mention)

Compiled URDSMMS User Survey Results

This appendix provides the compiled results of the above mentioned URDSMMS User Survey.

Survey Taken: 1/30/2005
Count: 10
Scale (1-5)

Question:	How would you rate your familiarity with the UniFrame project?
Total:	41
Average:	4.1

Question:	How would you rate your familiarity with the UniFrame Resource Discovery Service (URDS)?
Total:	41
Average:	4.1

Question:	How important is it to monitor the URDS?
Total:	50
Average:	5

Question:	How important is managing the URDS via an easy-to-use interface?
Total:	50
Average:	5

Question:	Do you think that the URDSMMS should contain two views, namely administrator view and system integrator/user view? Do you suggest any other view, if so please mention?
Yes:	10
No:	0

Rate the features by their importance, which you would like to, see in URDSMMS, on a scale of least important to very important

Question:

- Remote or local initiation and killing of different entities of URDS (such as Headhunters, Active Registries, Domain Security Manager and Query Manager)
  Total: 45
  Average: 4.5

- Submitting a query for searching components

Total:	46
Average:	4.6

- Representing the response time of a query in a graphical format with respect to Time

Total:	44
Average:	4.4

- Showing the statistical and individual details of all active entities of URDS in a single snap-shot

Total:	46
Average:	4.6

- Checking if the Headhunters are active or not

Total:	44
Average:	4.4

- Providing help for the URDSMMS

Total:	44
Average:	4.4

- Displaying the utilization of Headhunter or Query Manager at any point of time (so that we can know how much time the Headhunter or Query manager is busy)

Total:	44
Average:	4.4

User Interface Specific

Question:	Do you feel comfortable with a menu driven (menus and sub-menus) interface?
Yes:	10
No:	0

2. *URDSMMS Usability Testing Survey*

The below part of the appendix provides actual user survey mentioned in chapter 5. It consists of questions specific to the usability of the URDSMMS.

URDS Monitoring and Management System (URDSMMS)
Usability Testing Survey

- Was the interface of URDSMMS easy to navigate?
  (very hard) 1 2 3 4 5 (very easy)

- What do you think about the appearance of the interface?
  (strongly dislike) 1 2 3 4 5 (strongly like)

- How easy was it to get started using the system?
  (very hard) 1 2 3 4 5 (very easy)

- Was the help provided, useful in executing the features of URDSMMS?
  (Yes/No)

- How easy is it for a user to navigate independently through the system (after having executed it once before under supervision)?
  (very hard) 1 2 3 4 5 (very easy)

- Was the classification of two views (system administrator view and system integrator view) justified?
  Yes/No

- Was the interface able to convey the relevant information about each feature?
  (Yes/No)

- Thinking of the features and benefits of URDSMMS, rate your satisfaction?
  (very bad) 1 2 3 4 5 (excellent)

- If at all you were asked to build a monitoring system, do you expect to borrow any ideas from the URDSMMS.
  (Yes/No)

Compiled URDSMMS Usability Testing Survey Results

This appendix provides the compiled results of the above mentioned URDSMMS Usability Testing Survey.

Question:	Was the interface of URDSMMS easy to navigate?
Total:	29
Average:	4.833
Question:	What do you think about the appearance of the interface?
Total:	22
Average:	3.667
Question:	How easy was it to get started using the system?
Total:	27
Average:	4.5
Question:	Was the help provided, useful in executing the features of URDSMMS?
Yes:	6
No:	0
Question:	How easy is it for a user to navigate independently through the system (after having executed it once before under supervision)?
Total:	25
Average:	4.167
Question:	Was the classification of two views (system administrator view and system integrator view) justified?
Yes:	6
No:	0
Question:	Was the interface able to convey the relevant information about each feature?
Yes:	6
No:	0
Question:	Thinking of the features and benefits of URDSMMS, rate your satisfaction?
Total:	27
Average:	4.5
Question:	If at all you were asked to build a monitoring system, do you expect to borrow any ideas from the URDSMMS.
Yes:	6
No:	0

APPENDIX E: Class Diagram of URDSMMS Entities

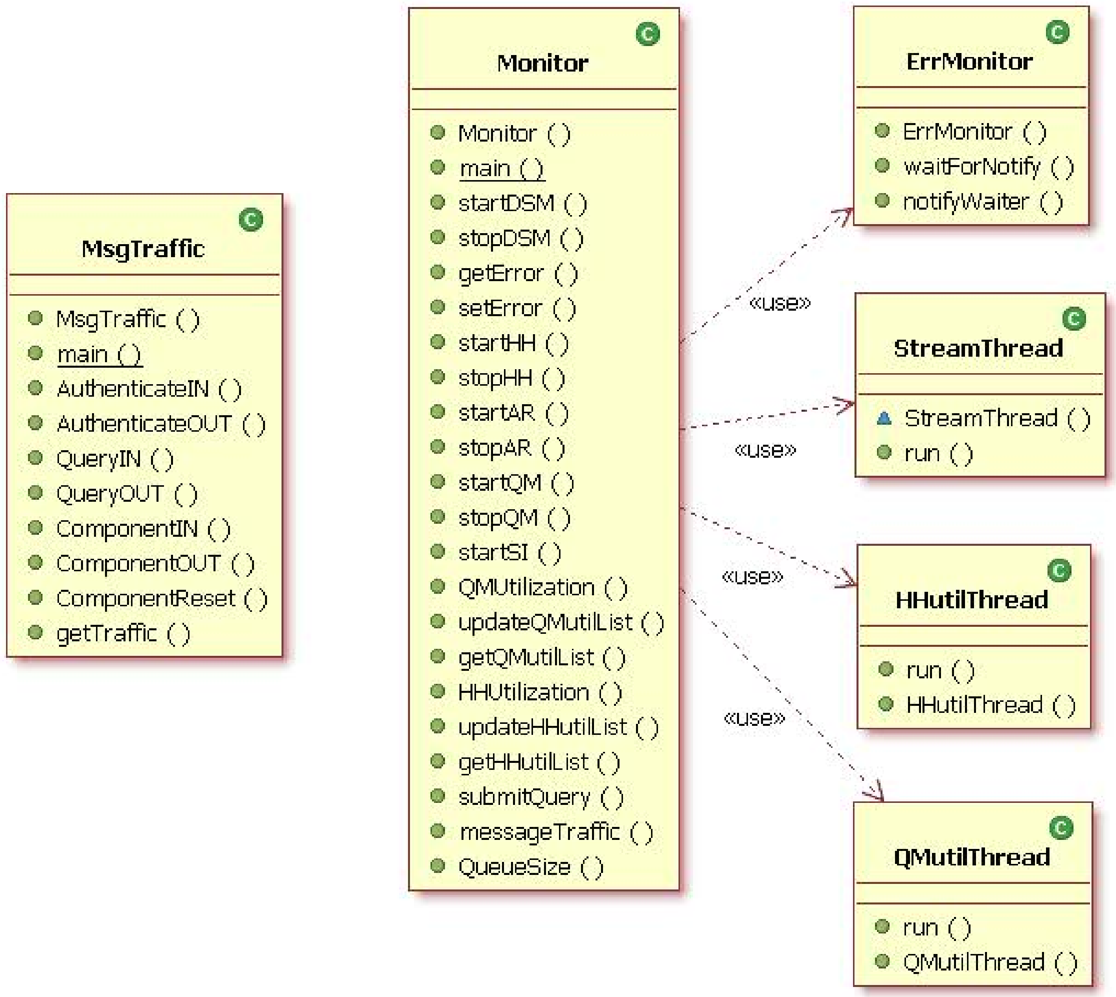


Figure: Class diagram of URDSMMS entities

APPENDIX F: Source Code

Monitor.java

```
/**
 * This class collects the monitoring information from URDS.
 *
 * @author Srikanth Reddy
 * @date Dec 2004
 */

import java.net.*;
import java.util.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
import java.lang.*;
import java.io.*;
import java.sql.*;

public class Monitor extends UnicastRemoteObject implements IMonitor
{
	private static Hashtable DSMprocessList = new Hashtable();
	private static Hashtable HHprocessList = new Hashtable();
	private static Hashtable ARprocessList = new Hashtable();
	private static Hashtable QMprocessList = new Hashtable();
	private static Hashtable SIprocessList = new Hashtable();
	// for storing utilization values
	public static ArrayList qmutList = new ArrayList();
	public static Hashtable StoreHHList = new Hashtable();

	private static int cnt = 0;
	private static StringBuffer errBuff1 = new StringBuffer();
	private IQueryManager utilQM = null;
	private IHeadhunter utilHH = null;
	private IURDS_Proxy1 urdsProxy = null;
	private IQueryManager qm = null;
	private IDomainSecurityManager dsm1 = null;
	private IMsgTraffic obj4 = null;


	public Monitor() throws RemoteException {
		super();
	}

	public static void main(String[] args)
	{
		String monitorLocation="//magellan.cs.iupui.edu:9879/Monitor";
		try
		{
			System.setSecurityManager(new RMISecurityManager());
			Naming.rebind(monitorLocation, new Monitor());
			System.out.println("Monitor " + monitorLocation + " is ready.");
		}catch(Exception e)
		{
			System.out.println("Monitor failed: " + e);
			errBuff1.append((String)e.getMessage());
		}
	}
```

```
		// to start DSM
		public int startDSM(String dsmlocation, int portNo)
		{
			int status = -1;
			Process p = null, p1 = null;
			StringBuffer outBuff3 = new StringBuffer();
			StringBuffer errBuff3 = new StringBuffer();
			StreamThread outThread3;
			StreamThread errThread3;
			BufferedInputStream stdOut3 = null;
			BufferedInputStream stdErr3 = null;

			try
			{
				Integer temp = new Integer(portNo);
				String fullLocn = "//" +dsmlocation +
".cs.iupui.edu:"+temp.toString()+"/DomainSecurityManager";
				System.out.println("Intial Time: " +System.currentTimeMillis());
				String cmd4 = "java -Xmx100m -Djava.security.policy=policy DomainSecurityManager
"+dsmlocation + " "+portNo;
				String cmd2 = "ssh -l sreddy phoenix java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy DomainSecurityManager "+dsmlocation + " "+portNo;
				ErrMonitor mon3= new ErrMonitor();
				if(dsmlocation.equalsIgnoreCase("magellan"))
				{
					p1 = Runtime.getRuntime().exec(cmd4);
					status = 0;
				}
				else if(dsmlocation.equalsIgnoreCase("phoenix"))
				{
					p1 = Runtime.getRuntime().exec(cmd2);
					status = 0;
				}

				if (status == 0)
				{
					DSMprocessList.put(fullLocn, p1);
					stdErr3 = new BufferedInputStream(p1.getErrorStream());
					stdOut3 = new BufferedInputStream(p1.getInputStream());
					errThread3 = new StreamThread(stdErr3, errBuff3, mon3);
					outThread3 = new StreamThread(stdOut3, outBuff3, mon3);
					errThread3.start();
					outThread3.start();
					Thread.sleep(15000);
					System.out.println("Process Completion Time: "
+System.currentTimeMillis());
					System.out.println("DSM Started");
				}
			}
			catch(Exception e)
			{
				System.out.println("Error...."+e.getMessage());
	e.printStackTrace();
				return(status);
			}
			return(status);
		}

		//to kill DSM
		public int stopDSM(String dsmlocation)
		{
        int dsm_status1 = -1;

        try
        {
            Enumeration e = DSMprocessList.keys();
            while(e.hasMoreElements())
            {
                String locn = (String) e.nextElement();
                System.out.println(" DSM process is " +locn+" and dsmlocn is "
+dsmlocation);
                if(locn.equalsIgnoreCase(dsmlocation))
                {
                    System.out.println(" Destroying DSM process " +dsmlocation);
                    Process p = (Process) DSMprocessList.get(locn);
                    p.destroy();
                    dsm_status1 = 0;
                    break;
                }
            }
            if (dsm_status1==0)
            {
                //to delete the dsmid from the database table
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                System.out.println("Database connected");
                Statement statement = dbconn.createStatement();
                String sqlstmt = null;
                sqlstmt = "delete from dsm where dsmid = '"+dsmlocation+"'";
                statement.executeQuery("delete from dsm where dsmid = '"+dsmlocation+"'");
                statement.close();
                dbconn.close();
                System.out.println("Query executed");
            }

        } catch(Exception ex)
        {
            ex.printStackTrace();
        }

        if (dsm_status1==0)
        {
            return(0);
        }
        else
            return(-1);
    }

    // to print error messages
    public StringBuffer getError() throws RemoteException
    {
        return errBuff1;
    }

    // to collect error messages
    public void setError(String error) throws RemoteException
    {
        if(error != null && error != "Headhunter")
        {
            errBuff1.append(error);
        }
```

```
   }

		// to start HH
		public int startHH(String hhlocation, int portNo, String dsmlocation, String domain, String name, String pwd)
		{
			int hh_status = -1;
			StringBuffer outBuff1 = new StringBuffer();
			StreamThread outThread1;
			StreamThread errThread1;
			BufferedInputStream stdOut1 = null;
			BufferedInputStream stdErr1 = null;

			Process p1= null, p2 = null;
			try
			{
				ErrMonitor mon1 = new ErrMonitor();
				Integer temp = new Integer(portNo);
				String fullLocn = "//" +hhlocation + ".cs.iupui.edu:"+temp.toString()+"/Headhunter";
				System.out.println("Intial Time: " +System.currentTimeMillis());
				String cmd4 = "java -Xmx100m -Djava.security.policy=policy Headhunter "+hhlocation
+ " "+portNo+ " " +dsmlocation+ " "+ domain+ " " +name+ " " +pwd;
				String cmd2 = "ssh -l sreddy phoenix java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy Headhunter "+hhlocation + " "+portNo+ " " +dsmlocation+ " "+
domain+ " " +name+ " " +pwd;

				System.out.println("Trying to start HH :"+hhlocation);
				if(hhlocation.equalsIgnoreCase("magellan"))
				{
					try
					{
						Runtime rt = Runtime.getRuntime();
						p1 = rt.exec(cmd4);
						hh_status = 0;
					}
					catch (Throwable t)
					{
						errBuff1.append((String)t.getMessage());
						t.printStackTrace();
					}
				}
				else if(hhlocation.equalsIgnoreCase("phoenix"))
				{
					p1 = Runtime.getRuntime().exec(cmd2);
					System.out.println("phoenix cmd2 executed");
					hh_status = 0;
				}
				else
				{
				String cmd3 = "ssh -l sreddy "+hhlocation+ " java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy Headhunter "+hhlocation + " "+portNo+ " " +dsmlocation+ " "+
domain+ " " +name+ " " +pwd;
					p1 = Runtime.getRuntime().exec(cmd3);
					System.out.println("cmd3 executed");
					hh_status = 0;
				}

				if (hh_status == 0)
				{
					HHprocessList.put(fullLocn, p1);
					stdErr1 = new BufferedInputStream(p1.getErrorStream());
					stdOut1 = new BufferedInputStream(p1.getInputStream());
```

```
                                        errThread1 = new StreamThread(stdErr1, errBuff1, mon1);
                                        outThread1 = new StreamThread(stdOut1, outBuff1, mon1);
                                        errThread1.start();
                                        outThread1.start();

                                        System.out.println("Process Completion Time: "
+System.currentTimeMillis());
                                        System.out.println("HH Started");
                                        //for maintaing the histroy of Utilization of HH
                                        HHUtilization(fullLocn);
                              }
                              else
                                        System.out.println("Error in starting the HH");

                    } catch(Exception e)
                    {
                              System.out.println("Error...."+e.getMessage());
        e.printStackTrace();
        errBuff1.append((String)e.getMessage());
                    }
                    return(hh_status);
          }

          // to kill HH
          public int stopHH(String hhlocation)
          {
                    int hh_status1 = -1;
                    Process p1 = null;
                    try
                    {
                              Enumeration e = HHprocessList.keys();

                              while(e.hasMoreElements())
                              {
                                        String locn = (String) e.nextElement();
                                        System.out.println(" HH process is " +locn+" and hhlocn is " +hhlocation);
                                        if(locn.equalsIgnoreCase(hhlocation))
                                        {
                                                  System.out.println(" Destroying HH process " +hhlocation);
                                                  p1 = (Process) HHprocessList.get(locn);
                                                  p1.destroy();
                                                  hh_status1 = 0;
                                                  break;
                                        }
                              }

                              // to remove the hh from the dsm list
                              dsm1 =
(IDomainSecurityManager)Naming.lookup("//magellan.cs.iupui.edu:2000/DomainSecurityManager");
                              dsm1.updateHHList(hhlocation);

                              if (hh_status1==0)
                              {
                                        //to delete the dsmid from the database table
                                        Class.forName("oracle.jdbc.driver.OracleDriver");
                                        Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                                        Statement statement = dbconn.createStatement();
                                        String sqlstmt = null;
                                        sqlstmt = "delete from hh where hhid = '"+hhlocation+"'";
                                        statement.executeQuery("delete from hh where hhid = '"+hhlocation+"'");
```

```
					statement.close();
					dbconn.close();
				}

			}catch(Exception ex1)
			{
				ex1.printStackTrace();
				errBuff1.append((String)ex1.getMessage());
			}

			return(hh_status1);
		}

		// to start AR
		public int startAR(String arlocation, int portNo, int coportNo, String dsmlocation, String domain, String 
name, String pwd)
		{
			int ar_status = -1;
			Process p1 = null, p2 = null;
			StringBuffer outBuff = new StringBuffer();
			StringBuffer errBuff = new StringBuffer();
			StreamThread outThread;
			StreamThread errThread;
			BufferedInputStream stdOut = null;
			BufferedInputStream stdErr = null;

			try
			{
				Integer temp = new Integer(portNo);
				String fullLocn = "//" +arlocation + ".cs.iupui.edu:"+temp.toString()+"/ActiveRegistry";
				System.out.println("Initial Time: " +System.currentTimeMillis());
				String cmd4 = "java -Xmx100m -Djava.security.policy=policy ActiveRegistry 
"+arlocation + " "+portNo+" " + coportNo+ " " +dsmlocation+ " "+ domain+ " " +name+ " " +pwd;
				String cmd2 = "ssh -l sreddy phoenix java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy ActiveRegistry "+arlocation + " "+portNo+ " "  + coportNo+ " " 
+dsmlocation+ " "+ domain+ " " +name+ " " +pwd;
				ErrMonitor mon = new ErrMonitor();

				System.out.println("Trying to start AR :"+arlocation);
				if(arlocation.equalsIgnoreCase("magellan"))
				{
				  try
					{
						System.out.println("printing ar cmd  "+cmd4);
						p1 = Runtime.getRuntime().exec(cmd4);
						ARprocessList.put(fullLocn, p1);
						ar_status = 0;
					}
					catch (Throwable t1)
					{
						t1.printStackTrace();
					}
				}
				else if(arlocation.equalsIgnoreCase("phoenix"))
				{
					p1 = Runtime.getRuntime().exec(cmd2);
					System.out.println("phoenix cmd2 executed");
					ar_status = 0;
				}
				else
				{
```

```
                        String cmd3 = "ssh -l sreddy "+arlocation+" java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy ActiveRegistry "+arlocation + " "+portNo+ " "  + coportNo+ " "
+dsmlocation+ " "+ domain+ " " +name+ " " +pwd;
                                p1 = Runtime.getRuntime().exec(cmd3);
                                System.out.println("cmd3 executed");
                                ar_status = 0;
                        }

                        if (ar_status == 0)
                        {
                                ARprocessList.put(fullLocn, p1);
                                stdErr = new BufferedInputStream(p1.getErrorStream());
                                stdOut = new BufferedInputStream(p1.getInputStream());
                                errThread = new StreamThread(stdErr, errBuff, mon);
                                outThread = new StreamThread(stdOut, outBuff, mon);
                                errThread.start();
                                outThread.start();
                                System.out.println("Process Completion Time: "
+System.currentTimeMillis());
                                System.out.println("AR Started");
                        }
                        else
                        {
                                System.out.println("Error in starting the AR");
                        }
                } catch(Exception e)
                {
                        System.out.println("Error...."+e.getMessage());
                        e.printStackTrace();
                }
                return(ar_status);
        }

        // to kill AR
        public int stopAR(String arlocation)
        {
                int ar_status1 = -1;

                try
                {
                        Enumeration e = ARprocessList.keys();

                        while(e.hasMoreElements())
                        {
                                String locn = (String) e.nextElement();
                                System.out.println(" AR process is " +locn+" and arlocn is " +arlocation);
                                if(locn.equalsIgnoreCase(arlocation))
                                {
                                        System.out.println(" Destroying AR process " +arlocation);
                                        Process p2 = (Process) ARprocessList.get(locn);
                                        p2.destroy();
                                        ar_status1 = 0;
                                        break;
                                }
                        }

                        dsm1 =
(IDomainSecurityManager)Naming.lookup("//magellan.cs.iupui.edu:2000/DomainSecurityManager");
                        dsm1.updateARList(arlocation);

                        if (ar_status1==0)
```

```
                                    {
                                                //to delete the dsmid from the database table
                                                Class.forName("oracle.jdbc.driver.OracleDriver");
                                                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                                                System.out.println("Database connected");
                                                Statement statement = dbconn.createStatement();
                                                String sqlstmt = null;
                                                sqlstmt = "delete from ar where arid = '"+arlocation+"'";
                                                statement.executeQuery("delete from ar where arid = '"+arlocation+"'");
                                                statement.close();
                                                dbconn.close();
                                                System.out.println("Query executed");
                                    }

                        } catch(Exception ex2)
                        {
                                    ex2.printStackTrace();
                        }

                        return(ar_status1);
            }

            // to start QM
            public int startQM(String qmlocation, int portNo, String dsmlocation,String name, String pwd)
            {
                        int qm_status = -1;
                        Process p1 = null, p2 = null;
                        StringBuffer outBuff2 = new StringBuffer();
                        StringBuffer errBuff2 = new StringBuffer();
                        StreamThread outThread2;
                        StreamThread errThread2;
                        BufferedInputStream stdOut2 = null;
                        BufferedInputStream stdErr2 = null;
                        try
                        {
                                    Integer temp = new Integer(portNo);
                                    String fullLocn = "//" +qmlocation + ".cs.iupui.edu:"+temp.toString()+"/QueryManager";
                                    System.out.println("Initial Time: " +System.currentTimeMillis());
                                    ErrMonitor mon2 = new ErrMonitor();
                                    String cmd4 = "java -Xmx100m -Djava.security.policy=policy QueryManager
"+qmlocation + " "+portNo+ " " +dsmlocation+ " "+name+ " " +pwd+" > qm.txt";
                                    String cmd2 = "ssh -l sreddy phoenix java -Xmx100m -
Djava.security.policy=/home/sreddy/UTest/policy QueryManager "+qmlocation + " "+portNo+ " " +dsmlocation+ "
"+name+ " " +pwd;

                                    System.out.println("Trying to start QM :"+qmlocation);
                                    if(qmlocation.equalsIgnoreCase("magellan"))
                                    {
                                                System.out.println("Printing status " +qm_status);
                                                p1 = Runtime.getRuntime().exec(cmd4);
                                                QMprocessList.put(fullLocn, p1);
                                                qm_status = 0;
                                                stdErr2 = new BufferedInputStream(p1.getErrorStream());
                                                stdOut2 = new BufferedInputStream(p1.getInputStream());
                                                errThread2 = new StreamThread(stdErr2, errBuff2, mon2);
                                                outThread2 = new StreamThread(stdOut2, outBuff2, mon2);
                                                errThread2.start();
                                                outThread2.start();
                                    }
                                    else
```

```
                    {
                              System.out.println("Invalid Location");
                    }
                    if (qm_status == 0)
                    {
                              System.out.println("Process Completion Time: "
+System.currentTimeMillis());
                              System.out.println("QM Started");
                              QMUtilization(fullLocn);
                    }
                    else
                                        System.out.println("Error in starting the QM");

                    } catch(Exception e)
                    {
                              System.out.println("Error...."+e.getMessage());
                              e.printStackTrace();
                    }
          return(qm_status);
}

// to kill QM
public int stopQM(String qmlocation)
{
          int qm_status1 = -1;
          try
          {
                    Enumeration e = QMprocessList.keys();

                    while(e.hasMoreElements())
                    {
                              String locn = (String) e.nextElement();
                              System.out.println(" QM process is " +locn+" and qmlocn is " +qmlocation);
                              if(locn.equalsIgnoreCase(qmlocation))
                              {
                                        System.out.println(" Destroying QM process " +qmlocation);
                                        Process p2 = (Process) QMprocessList.get(locn);
                                        p2.destroy();
                                        qm_status1 = 0;
                                        break;
                              }
                    }
                    if (qm_status1==0)
                    {
                              //to delete the dsmid from the database table
                              Class.forName("oracle.jdbc.driver.OracleDriver");
                              Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                              System.out.println("Database connected");
                              Statement statement = dbconn.createStatement();
                              String sqlstmt = null;
                              sqlstmt = "delete from qm where qmid = '"+qmlocation+"'";
                              statement.executeQuery("delete from qm where qmid = '"+qmlocation+"'");
                              statement.close();
                              dbconn.close();
                              System.out.println("Query executed");
                    }
          }catch(Exception ex3)
          {
                    ex3.printStackTrace();
          }
```

```
				return(qm_status1);
		}


		// to start System Integrator Proxy
		public int startSI()
		{
				int si_status = -1;
				int portNo = 2500;
				String qmlocation = "2000";
				Process p1 = null, p2 = null;
				StringBuffer outBuff5 = new StringBuffer();
				StringBuffer errBuff5 = new StringBuffer();
				StreamThread outThread5;
				StreamThread errThread5;
				BufferedInputStream stdOut5 = null;
				BufferedInputStream stdErr5 = null;
				try
				{
						Integer temp = new Integer(portNo);

						String fullLocn = "//magellan.cs.iupui.edu:"+temp.toString()+"/URDS_Proxy1";
						System.out.println("Initial Time: " +System.currentTimeMillis());
						ErrMonitor mon5 = new ErrMonitor();
						String cmd4 = "java -Xmx100m -Djava.security.policy=policy URDS_Proxy1
"+portNo+ " " +qmlocation;

						if(temp!=null && qmlocation!=null)
						{
								System.out.println("Trying to start System Integrator :"+fullLocn);
								System.out.println("Printing status " +si_status);
								p1 = Runtime.getRuntime().exec(cmd4);
								SIprocessList.put(fullLocn, p1);
								si_status = 0;
								stdErr5 = new BufferedInputStream(p1.getErrorStream());
								stdOut5 = new BufferedInputStream(p1.getInputStream());
								errThread5 = new StreamThread(stdErr5, errBuff5, mon5);
								outThread5 = new StreamThread(stdOut5, outBuff5, mon5);
								errThread5.start();
								outThread5.start();
						}

						if (si_status == 0)
						{
								System.out.println("Process Completion Time: "
+System.currentTimeMillis());
								System.out.println("System Integrator Started");
						}
						else
								System.out.println("Error in starting the System Integrator");

				} catch(Exception e)
				{
						System.out.println("Error...."+e.getMessage());
						e.printStackTrace();
				}
				return(si_status);
		}
```

```
	//For starting Utlization Thread for QM
	public void QMUtilization(String qmLocation)
	{
		// Thread for HHutilThread
		try
		{
			QMutilThread qmThreadObj = new QMutilThread(this, qmLocation);
			Thread qmThread = new Thread(qmThreadObj);

			Thread.sleep(10000);
			qmThread.start();
		}
		catch (Exception e1)
		{
			System.out.println("\n#### ERROR in QMUtilThread Starting: " + e1.getMessage());
			e1.printStackTrace();
		}
	}
	//to update the qmutList for QM
	public void updateQMutilList(float hhUper)
	{
		Float ff = new Float(hhUper);
		if(ff!=null)
		{
			qmutList.add(ff);
		}
	}

	//called to retrieve the LinkedList of the util%
	public ArrayList getQMutilList()
	{
		return qmutList;
	}

	//For starting Utilization Thread for each HH
	public void HHUtilization(String hhLocation)
	{
		// Thread for HHutilThread
		try
		{
			HHutilThread hhThreadObj = new HHutilThread(this, hhLocation);
			Thread hhThread = new Thread(hhThreadObj);

			Thread.sleep(10000);
			hhThread.start();
			ArrayList hhutList = new ArrayList();
			StoreHHList.put(hhLocation, hhutList);
		}
		catch (Exception e)
		{
			System.out.println("\n#### ERROR in HHUtilThread Starting: " + e.getMessage());
			e.printStackTrace();
		}
	}

	//to update the hhutList for each HH
	public void updateHHutilList(String hhID, float hhUper)
	{
		Float ff = new Float(hhUper);
		Enumeration e = StoreHHList.keys();
```

```
			ArrayList hhutList1 = new ArrayList();
			while(e.hasMoreElements())
			{
				String id = (String) e.nextElement();
				if(id.equals(hhID))
				{
					hhutList1 = (ArrayList)StoreHHList.get(id);
					hhutList1.add(ff);
					break;
				}
			}
		}

		//called to retrieve the LinkedList of the util%
		public ArrayList getHHutilList(String hhLoc)
		{
			ArrayList hhutList2 = new ArrayList();
			Enumeration e1 = StoreHHList.keys();
			String id1 = "";

			while(e1.hasMoreElements())
			{
				id1 = (String) e1.nextElement();
				if(id1.equals(hhLoc))
				{
					hhutList2 = (ArrayList)StoreHHList.get(id1);
					break;
				}
			}
			return hhutList2;
		}

		// to submit a query for processing
		public int submitQuery(String cName, long reTime)
		{
			int stat = 0;
			try
			{
				System.out.println("before lookup URDS Proxy1");
				String proxyLocation = "//magellan.cs.iupui.edu:2500/URDS_Proxy1";
				urdsProxy = (IURDS_Proxy1) Naming.lookup(proxyLocation);
				System.out.println("URDS Proxy1");
				urdsProxy.searchConcreteComponents(cName, reTime);
			}
			catch(Exception ex6)
			{
				ex6.printStackTrace();
				return(-1);
			}
			return (stat);

		}

		// to collect message traffic information
		public ArrayList messageTraffic()
		{
			ArrayList list1 = new ArrayList();

			try
			{
				obj4 = (IMsgTraffic)Naming.lookup("//magellan.cs.iupui.edu:9880/MsgTraffic");
```

```
                    list1.clear();
                    list1 = obj4.getTraffic();
                }
                catch (Exception ex7)
                {
                    System.out.println("Error in messageTraffic"+ex7.getMessage());
                }
                return list1;
        }

        // find the no. of processing
        public int QueueSize()
        {
                int i = 0;
                try
                {
                    qm = (IQueryManager)Naming.lookup("//magellan.cs.iupui.edu:2000/QueryManager");
                    i = qm.queueSize();
                }
                catch (Exception ex8)
                {
                    System.out.println("Error in QueueSize"+ex8.getMessage());
                }
                return i;
        }
}
```

MsgTraffic.java

```
/**
 * This class collects the traffic information of the URDS.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.net.*;
import java.util.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
import java.lang.*;
import java.io.*;
import java.sql.*;

public class MsgTraffic extends UnicastRemoteObject implements IMsgTraffic
{
        private static Integer auth = new Integer(0);
        private static Integer query = new Integer(0);
        private static Integer comp = new Integer(0);
        private static int a=0, b=0, c=0;

        public MsgTraffic() throws RemoteException {

                // Thread for MsgTrafficThread
                try
                {
                    MsgTrafficThread ThreadObj = new MsgTrafficThread(this);
                    Thread TraThread = new Thread(ThreadObj);

                    Thread.sleep(10000);
                TraThread.start();
            }
            catch (Exception e1)
            {
                System.out.println("\n#### ERROR in MsgTrafficThread Starting: " + e1.getMessage());
                e1.printStackTrace();
            }

    }

    public static void main(String[] args)
    {
            String msgTrafficLocation="//magellan.cs.iupui.edu:9880/MsgTraffic";
            try
            {
                System.setSecurityManager(new RMISecurityManager());
                Naming.rebind(msgTrafficLocation, new MsgTraffic());
                System.out.println("MsgTraffic " + msgTrafficLocation + " is ready.");
            }catch(Exception e)
            {
                System.out.println("MsgTraffic failed: " + e);
            }
    }

    public void AuthenticateIN()
    {
            synchronized(auth)
            {
                a++;
            }
            System.out.println("increment value of a "+a);
            return;
    }

    public void AuthenticateOUT()
    {
            synchronized(auth)
            {
                a--;
            }
            System.out.println("decrement value of a "+a);
            return;
    }

    public void QueryIN()
    {
            synchronized(query)
            {
                b++;
            }
            System.out.println("increment value of b "+b);
            return;
    }

    public void QueryOUT()
    {
            synchronized(query)
            {
                b--;
            }
            System.out.println("decrement value of b "+b);
```

```
                        return;
            }

            public void ComponentIN()
            {
                        synchronized(comp)
                        {
                                    c++;
                        }
                        System.out.println("increment value of c "+c);
                        return;
            }

            public void ComponentOUT()
            {
                        synchronized(comp)
                        {
                                    c--;
                        }
                        System.out.println("decrement value of c "+c);
                        return;
            }

            //will be called by Monitor.java
            public ArrayList getTraffic()
            {
                        ArrayList list = new ArrayList();

                        list.clear();

                        Integer a1 = new Integer(a);
                        Integer b1 = new Integer(b);
                        Integer c1 = new Integer(c);

                        list.add(a1);
                        list.add(b1);
                        list.add(c1);

                        return list;
            }
}
```

HHutilThread.java

```
/**
 * This thread periodically computes the HH utilization percentage.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.security.*;
import java.util.*;
import java.lang.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;

public class HHutilThread implements Runnable
{
   private IHeadhunter utilHH = null;
```

```
            private Monitor mm = null;
    private String hhLocation = "";
    private long sleepTime = 60000;                         // Time for which the thread

    public void run()
    {
                    Thread CurrentThread = Thread.currentThread();

                    double util_netTime1 = 0;
                    double util_stopTime1 = 0;
                    float utilper1 = -1;
                    double util_startTime1 = 0;
                    try
                    {
                            CurrentThread.sleep(20000);
                            System.out.println("printing hhLocation in HHutilThread "+ hhLocation);
                            utilHH = (IHeadhunter)Naming.lookup(hhLocation);
                    }
                    catch(Exception e1)
                    {
                            System.out.println("error in lookup HH "+e1.getMessage());
                            try
                            {
                                    mm.setError("Error in HHUtilLookup " +e1.getMessage());
                            }
                            catch(Exception e)
                            {
                                    System.out.println("Caught some exception ");
                            }
                    }

                    while(true)
                    {
                            util_netTime1 = 0;
                            util_stopTime1 = 0;
                            util_startTime1 = 0;
                            try
                            {
                                    util_startTime1 = System.currentTimeMillis();
                                    utilHH.HHutilStart();
                                    System.out.println("Printing Start Time of HH " + util_startTime1);
                                    // sleep for some time to record the utilization of HH
                                    CurrentThread.sleep(sleepTime);
                                    util_netTime1 = utilHH.HHutilStop();
                                    util_stopTime1 = System.currentTimeMillis();
                                    double totalTime1 = util_stopTime1 - util_startTime1;
                                    utilper1 = (float)(util_netTime1 / totalTime1) * 100 ;
                                    System.out.println("Printing Utilization Percentage of HH " + utilper1);

                                    if(utilper1<=100)
                                    {
                                            // to update the table in Monitor
                                            mm.updateHHutilList(hhLocation, utilper1);
                                    }
                            }
                            catch(Exception ex5)
                            {
                                    try
                                    {
                                            mm.setError("Error in HHUtilLookup loc2 "+ex5.getMessage());
                                    }
```

```
                                                catch(Exception e)
                                                {
                                                            System.out.println("error here in hhutil "+e.getMessage());
                                                }
                                                System.out.println("Caught some exception ");
                                    }
                        }
    }

    public HHutilThread(Monitor mon, String hhLoc)
    {
                        try
                        {
                                    mm = mon;
                                    hhLocation = hhLoc;
                        }
                        catch (Exception e)
                        {
                           System.out.println("#### ERROR in HHutilThread constructor: " + e);
                           e.printStackTrace();
                           System.exit(1);
                        }
    }
}
```

QMutilThread.java

```
/**
 * This thread periodically computes the QM utilization percentage.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.security.*;
import java.util.*;
import java.lang.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;

public class QMutilThread implements Runnable
{
    private IQueryManager utilQM = null;
             private Monitor mm1 = null;
    private String qmLocation = "";
    private long sleepTime = 60000;                              // Time for which the thread

    public void run()
    {
                        Thread CurrentThread = Thread.currentThread();

                        double util_netTime = 0;
                        double util_stopTime = 0;
                        float utilper = -1;
                        double util_startTime = 0;
                        try
                        {
                                    System.out.println("printing qmLocation "+ qmLocation);
                                    utilQM = (IQueryManager)Naming.lookup(qmLocation);
```

```
                }
                catch(Exception e2)
                {
                        System.out.println("error in lookup "+e2.getMessage());
                }

                while(true)
                {
                        util_netTime = 0;
                        util_stopTime = 0;
                        util_startTime = 0;
                        try
                        {
                                util_startTime = System.currentTimeMillis();
                                utilQM.QMutilStart();
                                System.out.println("Printing Start Time of QM " + util_startTime);

                                // sleep for some time to record the utilization of QM
                                CurrentThread.sleep(sleepTime);
                                util_netTime = utilQM.QMutilStop();
                                util_stopTime = System.currentTimeMillis();
                                System.out.println("Printing Stop Time of QM " + util_stopTime);
                                double totalTime = util_stopTime - util_startTime;
                                System.out.println("Printing Net Time of QM " + util_netTime);
                                utilper = (float)(util_netTime / totalTime) * 100 ;
                                System.out.println("Printing Utilization Percentage of QM " + utilper);

                                if(utilper<=100)
                                {
                                        // to update the table in Monitor
                                        mm1.updateQMutilList(utilper);
                                }
                        }
                        catch(Exception ex5)
                        {
                                try
                                {
                                        mm1.setError(ex5.getMessage());
                                }
                                catch(Exception e1)
                                {
                                        System.out.println("Caught some exception ");
                                }
                                ex5.printStackTrace();
                        }
                }
}

public QMutilThread(Monitor mon, String qmLoc)
{
                try
                {
                        mm1 = mon;
                        qmLocation = qmLoc;
                }
                catch (Exception e)
                {
                  System.out.println("#### ERROR in QMutilThread constructor: " + e);
                  e.printStackTrace();
                  System.exit(1);
                }
```

```
    }
}
```

StreamThread.java

```
/**
 * This class stores the monitoring information in a StringBuffer.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.io.*;
import java.util.*;

public class StreamThread extends Thread
{
		BufferedInputStream stream;
		StringBuffer buff;
		ErrMonitor mon;
		StreamThread(BufferedInputStream in, StringBuffer buf, ErrMonitor m)
		{
			stream = in;
			buff = buf;
			mon = m;
		}
		public void run()
		{
			try
			{
				int i;
				yield();
				while((i = stream.read()) > -1)
				{
					buff.append((char)i);
				}
				mon.notifyWaiter();
			}
			catch (Exception ioe)
			{
				System.out.println("thread exception "+ioe.getMessage());
				ioe.printStackTrace();
			}
		}
}
```

ErrMonitor.java

```
/**
 * This class collects the error messages.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.io.*;

class ErrMonitor
{
		public ErrMonitor()
		{}

		synchronized public void waitForNotify()
		{
				try
				{
						wait();
				}
				catch(InterruptedException e) {}
		}
		synchronized public void notifyWaiter()
		{
				notify();
		}
}
```

RMIHelper.java

```
/**
 * This is a helper class between the JSP pages and the RMI classes.
 *
 * @author Srikanth Reddy
 * @date Jan 2005
 */

import java.util.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;

public class RMIHelper {

 private IMonitor monitor = null;

 public RMIHelper()
 {
   try
   {
		String monitorlocation ="//magellan.cs.iupui.edu:9879/Monitor";
		monitor = (IMonitor) Naming.lookup(monitorlocation);
   } catch (Exception e)
   {
		System.out.println(e.getMessage());
   }
 }
 public int startDSM(String dsmaddress, int portNo)
 {
    int status = -1;
    try
    {
		status = monitor.startDSM(dsmaddress, portNo);

    } catch (Exception e1)
    {
		System.out.println(e1.getMessage());
    }
    return(status);
}
```

```
public int startHH(String hhaddress, int portNo, String dsmaddress, String domain, String name, String pwd)
{
        int hh_status = -1;
        try
        {
                hh_status = monitor.startHH(hhaddress, portNo, dsmaddress, domain, name, pwd);
        }catch (Exception e2)
        {
                System.out.println(e2.getMessage());
        }
        return(hh_status);
}

public int startAR(String araddress, int portNo, int coportNo, String dsmaddress, String domain, String name, String
pwd)
{
        int ar_status = -1;
        try
        {
                ar_status = monitor.startAR(araddress, portNo, coportNo, dsmaddress, domain, name, pwd);
        }catch (Exception e3)
        {
                System.out.println(e3.getMessage());
        }
        return(ar_status);
 }

 public int startQM(String qmaddress, int portNo, String dsmaddress, String name, String pwd)
 {
        int qm_status = -1;
        try
        {
                qm_status = monitor.startQM(qmaddress, portNo, dsmaddress, name, pwd);
        }catch (Exception e4)
        {
                System.out.println(e4.getMessage());
        }
        return(qm_status);
 }

 public int startSI()
 {
        int si_status = -1;
        try
        {
                si_status = monitor.startSI();
        }catch (Exception e13)
        {
                System.out.println(e13.getMessage());
        }
        return(si_status);
 }

public int stopDSM(String dsmlocn)
{
        int dsm_status = -1;
        try
        {
                dsm_status = monitor.stopDSM(dsmlocn);
        }catch (Exception e5)
        {
                System.out.println(e5.getMessage());
        }
        return(dsm_status);
}

public int stopHH(String hhlocn)
{
        int hh_status = -1;
        try
        {
                hh_status = monitor.stopHH(hhlocn);
        }catch (Exception e6)
        {
                System.out.println(e6.getMessage());
        }
        return(hh_status);
}

public int stopAR(String arlocn)
{
        int ar_status = -1;
        try
        {
                ar_status = monitor.stopAR(arlocn);
        }catch (Exception e7)
        {
                System.out.println(e7.getMessage());
        }
        return(ar_status);
}

public int stopQM(String qmlocn)
{
        int qm_status = -1;
        try
        {
                qm_status = monitor.stopQM(qmlocn);
        }catch (Exception e8)
        {
                System.out.println(e8.getMessage());
        }
        return(qm_status);
}

public StringBuffer getError()
{
         try
         {
                 StringBuffer errorbuf = (StringBuffer)monitor.getError();
                 return errorbuf;
         }
         catch(Exception e)
         {
                return null;
         }

}

public ArrayList QMUtil()
{
  ArrayList upercent_qm = new ArrayList();
```

```
    try
    {
       upercent_qm = monitor.getQMutilList();

    }catch (Exception e9)
    {
       System.out.println(e9.getMessage());
    }
    return(upercent_qm);
}

public ArrayList HHUtil(String hhLocn)
{
    ArrayList upercent_hh = new ArrayList();
    try
    {
       upercent_hh = monitor.getHHutilList(hhLocn);

    }catch (Exception e10)
    {
       System.out.println(e10.getMessage());
    }
    return upercent_hh;
}

public int subQuery(String compName, long respTime)
{
        int stat = -1;
        try
        {
                stat = monitor.submitQuery(compName, respTime);
        }catch (Exception e11)
        {
                System.out.println(e11.getMessage());
        }
        return(stat);
}

public ArrayList msgTraffic()
{
        ArrayList list2 = new ArrayList();
        try
        {
                list2 = monitor.messageTraffic();
        }catch(Exception e12)
        {
                System.out.println(e12.getMessage());
        }
        return list2;
}

public int queSize()
{
        int k = 0;
        try
        {
                k = monitor.QueueSize();
        }catch(Exception e14)
        {
                System.out.println(e14.getMessage());
        }
```

```
                return k;
 }
}
```

URDS_Proxy1.java

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.net.*;
import java.util.*;
import java.sql.*;

/**
 * This class implements the URDS_Proxy in URDSMMS to interface with URDS.
 *
 * @author Barun Devaraju
 * @modified Srikanth Reddy
 * @date Dec 2004
 */

public class URDS_Proxy1 extends UnicastRemoteObject implements IURDS_Proxy1
{
   private boolean timedExperiment = false;

   private IQueryManager queryManager = null;
   AbstractComponent component = new AbstractComponent();
   private QueryBean queryBean = null;
   private long responseTime = 400000;
   private int totalQueries=1; // total no of queries sent at once
   private int queriesSent=0;
   private int resultsReceived=0;
   private int experimentNumber = 1; // no of times the experiment is repeated
   private int experimentDone = 0;
   private String domainName = "Document";
   private String location = null;
   private Timer timer = null;
   private long interval = 200;
            private IMsgTraffic obj2 = null;
            private IMonitor mon = null;

   public URDS_Proxy1(String qmPort, String url) throws RemoteException
   {
      try
      {
            location = url;
            queryManager =(IQueryManager)Naming.lookup("//magellan.cs.iupui.edu:" + qmPort + "/QueryManager");
            obj2 = (IMsgTraffic)Naming.lookup("//magellan.cs.iupui.edu:9880/MsgTraffic");
            mon = (IMonitor)Naming.lookup("//magellan.cs.iupui.edu:9879/Monitor");
            }
      catch(RemoteException e)
      {
         System.err.println(e);
      }
      catch(NotBoundException e)
      {
         System.err.println(e);
      }
      catch(MalformedURLException e)
      {
          System.err.println(e);
        }

    }

              // to submit new query
    public void newQuery(String compName, long respTime)
    {
              domainName = compName;
              responseTime = respTime;
        try
        {
          System.out.println("in new Query URDS_Proxy1");
          this.searchConcreteComponents(domainName, responseTime);
        }
        catch(RemoteException e)
        {
          System.err.println(e.getMessage());
        }
              }

    public void searchConcreteComponents(String coName, long resTime) throws RemoteException
    {
        domainName = coName;
        responseTime = resTime;

        component.setDomainName(domainName);
              // to search for document server only query through URDS
        if(queryManager == null)
        {
          System.out.println("Query Manager is not ready");
        }
                        else
                        {
                          queryBean = new QueryBean(component);
                          queriesSent = 0;
                          resultsReceived = 0;
                          if(timedExperiment)
                          {
                                  //System.out.println("Timer scheduled for interval " + interval);
                          }
                          else
                          {
                                  while(queriesSent < totalQueries)
                                  {
                                   try
                                      {
                                          obj2.QueryIN();
                                          queryManager.getSearchResultTable(queryBean, responseTime, location);
                                          queriesSent++;
                                          System.out.println(">>>> Query " + queriesSent + " sent at "
 +System.currentTimeMillis());
                        }
                                    catch(Exception e )
                                    {
                                          try
                                                  {
                                                          mon.setError(e.getMessage());
                                                  }
                                                  catch(Exception e1)
                                                  {
```

```
                                                                System.out.println("Caught some exception ");
                                                        }
                                                System.out.println("#### ERROR in contacting Query Manager and getting
results ");
                                        }
                                }
                        System.out.println("Totally " + queriesSent + " queries sent");
                }
        }
  }


  // Called when the results are available
  public int obtainResults(String queryID, Hashtable result)
  {
                try
                {
                        obj2.QueryOUT();
                }
                catch(Exception e)
                {
                        System.out.println(e.getMessage());
                }

                resultsReceived++;
                System.out.println("URDS_Proxy1: Result Size - " + result.size());
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                        System.out.println("Database connected");
                        Statement statement = dbconn.createStatement();

                        if(result != null)
                        {
                                Enumeration values = result.keys();
                                while(values.hasMoreElements())
                                {
                                        String componentID = (String)values.nextElement();
                                        ComponentDetailsObject cdo = (ComponentDetailsObject)
result.get(componentID);
                                        System.out.println(componentID
+"\t"+cdo.getConcreteComponent().getComponentName()+"\t"+cdo.getComponentRating());
                                        statement.executeQuery("insert into urdsresults values
('"+queryID+"', '"+componentID+"','"+cdo.getConcreteComponent().getComponentName()+"',
"+cdo.getComponentRating()+" )");
                                 }

                                if(result.size()==0)
                                {
                                        System.out.println(" ----- No component matched for Query " +
queryID);
                                }

                                System.out.println("URDS Proxy obtanied results for the query " + queryID);
                                System.out.println("------------- END --------------- " + resultsReceived + " " +
queriesSent + " " + totalQueries +"  "+System.currentTimeMillis());
                        }
                        obj2.QueryIN();
                        statement.close();
                        dbconn.close();
                    }
                    catch (Exception e1)
                    {
                        try
                        {
                            mon.setError(e1.getMessage());
                        }
                        catch(Exception e2)
                        {
                            System.out.println("Caught some exception ");
                        }

                        System.out.println("Exception at obtainresults");
                    }

                    return 0;
    }

    public static void main(String[] args)
    {
        String url = null;
        String qmPort = null;

      if(args.length > 2)
      {
        System.out.println("Usage: java URDS_Proxy1 server_name responseTime");
      }

      if(args.length == 0)
        {
          url = "//magellan.cs.iupui.edu:2500/URDS_Proxy1";
        }
        else
        {
          url = "//magellan.cs.iupui.edu:" + args[0] + "/URDS_Proxy1";
        }

      if(args.length == 1)
      {
        qmPort = "2000";
      }
      else
      {
        qmPort = args[1];
      }

      try
      {
        URDS_Proxy1 urds_Proxy= new URDS_Proxy1(qmPort, url);
        Naming.rebind(url, urds_Proxy);
          System.out.println("URDS_Proxy1 is registered in location " + url);
          System.out.println("URDS Proxy started");
      }
      catch(RemoteException e)
      {
        System.err.println(e.getMessage());
      }
      catch(MalformedURLException e)
      {
        System.err.println(e.getMessage());
        }
        catch(Exception e)
        {
           System.err.println(e);
           System.exit(1);
        }
    }
}
```

JSP Code

main.jsp

```
<html>

<head>
<title> URDS Monitoring and Management System </title>
<link rel="stylesheet" href="web.css" type="text/css">
<base target="_self">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td align="center"><a href="http://www.cs.iupui.edu/uniFrame" target="_blank">
        <img src="images/uniFrameHeader.gif" width= "800" height="147" border="0"></a></td>
    </tr>
     <tr>
       <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
     </tr>
</table>

<p>
<font face="Arial, Helvetica, sans-serif"><b>
<h3>
<center>
<%!
   int first = 1;
%>
<%
   Boolean authenflag = (Boolean) request.getAttribute("authenflag");
   if(authenflag == null ||first == 1)
   {
     first = 0;
   }
   else
   {
%>
    <font color =red> Authentication failed. Please re-enter User name and Password. </font>
<%
   }
%>
</center>
</h3>
</p>
<form  name="MainPage" action="process2.jsp" method = "post" >
```

```
 <table border="1" bgcolor="#C0C0C0">
  <tr>
  <td>
     Username
  </td>
  <td align="center">
     <input type="text" name="username">
  </td>
  </tr>
<tr>
  <td>
     Password
  </td>
  <td align="center">
     <input type="password" name="password">
  </td>
</tr>
<br>
</table>
<br>
<br>
<input type="submit" name="MainPage" value="Login">
 <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/help_mainPage.htm">
 <font face="Arial" color="#0000FF">help?</font></a>
</form>

<br>
<table>
    <tr>
      <td><a href="http://www.onr.navy.mil/sci_tech/" target="_blank">
      <img src="images/sponsor_logos.jpg" width= "577" height="80" border="0"></a></td>
    </tr>
</table>
</center>
</body>
</html>
```

process2.jsp

```
<%@ page import="java.util.*, java.lang.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<jsp:useBean id="idHandler" class="security.Login" scope="request">
<jsp:setProperty name="idHandler" property="*"/>
</jsp:useBean>

<%  if (idHandler.authenticate((String)request.getParameter("username"), (String)request.getParameter("password")))
    {
            if(request.getParameter("MainPage") != null)
            {
              String st1 = (String) request.getParameter("username").trim();
              if(st1.equals("admin"))
              {
%>
                      <jsp:forward page="first.htm"/>
<%            }
              else if(st1.equals("user"))
              {
%>
                      <jsp:forward page="first_si.htm"/>
<%
```

```
                }
            }
   }
   else
   {
       Boolean authenflag = new Boolean("true");
       request.setAttribute("authenflag", authenflag);
%>
       <jsp:forward page="main.jsp"/>
<%
   }
%>
```

Monitor.jsp

```
/**
 * This JSP page collects the statistical details of the URDS.
 *
 * @author Srikanth Reddy
 * @date Feb 2005
 */

<%@ page language="java" import="java.sql.*, RMIHelper" %>
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<%
            RMIHelper obj2 = new RMIHelper();
            int j = 0;
            j = obj2.queSize();
%>

<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #F8F7D9>

<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            Statement statement = dbconn.createStatement();
%>
  <table width="550" border="0" cellspacing="1" cellpadding="0">
    <tr>
            <td bgcolor="#800000" height="30" colspan="5" align="center" bordercolor="#C0C0C0" >
            <font color="#FFFFFF" size="5">
            <b> Global Snapshot of URDS </b>
            </font>
            </td>
    </tr>
    <%
            ResultSet rst = statement.executeQuery("select * from dsm");
            String dsmval="";
            String dsmname = "Domain Security Manager";
    %>
      <tr>
            <td height="28">
            <B><%=dsmname%> </B>
            </td>
    <%
```

```
            while(rst.next())
            {
                dsmval=rst.getString("DSMID");
            }
%>
            <td height="28">
            <% if (dsmval != "" )
            { %>

            <B><%=dsmval%></B>

            <% }
            else
            { %>
            <B> No Active DSM </B>
            <% } %>
            </td>
</tr>

<br>

<%
            ResultSet rst1 = statement.executeQuery("select * from qm");
            String qmval="";
            String qmname = "Query Manager";

   while(rst1.next())
            {
                qmval=rst.getString("QMID");
            }
 %>
<tr>
   <td height = "28">
   <B><%=qmname%> </B>
   </td>

            <td height = "28">
            <% if (qmval != "" )
            { %>
            <B><%=qmval%></B>

            <% }
            else
            { %>
            <B> No Active QM </B>
            <% } %>
            </td>
</tr>


<%
            ResultSet rst2 = statement.executeQuery("SELECT COUNT(*) as HHNO FROM HH");
            int hhno= 0;
            String hhname = "No of Headhunters Active";

            while(rst2.next())
            {
                 hhno = rst2.getInt("HHNO");
            }
%>
```

```
<tr>
        <td height = "28">
        <B><%=hhname%> </B>
        </td>

        <td height = "28">
        <% if (hhno != 0 )
        { %>
        <B><%=hhno%></B>

        <% }
        else
        { %>
        <B> No Active HHs </B>
        <% } %>
        </td>
</tr>
<tr>
        <td>
        <b>
        <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor_hh_details.jsp">
        Display Details</a>
        </b>
        </td>
</tr>

<%
        ResultSet rst3 = statement.executeQuery("SELECT COUNT(*) as ARNO FROM AR");
        int arno= 0;
        String arname = "No of Active Registries Active";

        while(rst3.next())
        {
            arno = rst3.getInt("ARNO");
        }
%>

<tr>
        <td height = "19">
        <B><%=arname%> </B>
        </td>

        <td height = "19">
        <% if (arno != 0 )
        { %>
        <B><%=arno%></B>

        <% }
        else
        { %>
        <B> No Active ARs </B>
        <% } %>
        </td>
</tr>
<tr>
        <td>
        <b>
        <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor_ar_details.jsp">
        Display Details</a>
        </b>
        </td>
```

```
   </tr>
   <tr>
              <td height = "19">
              <B> No. of Queries being processed: </B>
              </td>
              <td height = "19">
              <B> <%=j%> </B>
              </td>
   </tr>
<%
              statement.close();
              dbconn.close();
%>
</table>
</body>
</html>
```

Monitor_hh_details.jsp

```
/**
 * This JSP page collects the details of all the HHs.
 *
 * @author Srikanth Reddy
 * @date Feb 2005
 */

<%@ page language="java" import="java.sql.*, RMIHelper" %>
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
              RMIHelper obj2 = new RMIHelper();
              int j = -1;
              j = obj2.queSize();
%>

<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #F8F7D9>

<%
              Class.forName("oracle.jdbc.driver.OracleDriver");
              Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
              Statement statement = dbconn.createStatement();
%>
  <table width="550" border="0" cellspacing="1" cellpadding="0" height="84">
    <tr>
              <td bgcolor="#800000" height="30" colspan="5" align="center" bordercolor="#C0C0C0" >
              <font color="#FFFFFF" size="5">
              <b> Global Snapshot of URDS </b>
              </font>
              </td>
    </tr>
    <%
              ResultSet rst = statement.executeQuery("select * from dsm");
              String dsmval="";
              String dsmname = "Domain Security Manager";
```

```
        while(rst.next())
        {
            dsmval=rst.getString("DSMID");
        }
%>
<tr>
        <td height="28" width="333">
        <B><%=dsmname%> </B>
        </td>

        <td height="28" width="364">
        <% if (dsmval != "" )
        { %>

        <B><%=dsmval%></B>

        <% }
        else
        { %>

        <B> No Active DSM </B>

        <% } %>
        </td>
</tr>

<br>
<br>

<%
        ResultSet rst1 = statement.executeQuery("select * from qm");
        String qmval="";
        String qmname = "Query Manager";

        while(rst1.next())
        {
            qmval=rst.getString("QMID");
        }
%>
<tr>
        <td height = "28" width="333">
        <B><%=qmname%> </B>
        </td>

        <td height = "28" width="364">
        <% if (qmval != "" )
        { %>
        <B><%=qmval%></B>

        <% }
        else
        { %>
        <B> No Active QM </B>
        <% } %>
        </td>
</tr>


<%
        ResultSet rst2 = statement.executeQuery("SELECT COUNT(*) as HHNO FROM HH");
        int hhno= 0;
```

```
            String hhname = "No of Headhunters Active";

            while(rst2.next())
            {
                hhno = rst2.getInt("HHNO");
            }
    %>

    <tr>
            <td height = "28" width="333">
            <B><%=hhname%> </B>
            </td>

            <td height = "28" width="364">
            <% if (hhno != 0 )
            { %>
            <B><%=hhno%></B>

            <% }
            else
            { %>
            <B> No Active HHs </B>
            <% } %>
            </td>
    </tr>

   <tr>
            <td>
            <b>
            <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor.jsp">
            Hide Details</a>
            </b>
            </td>
   </tr>
  </table>
    <%
            ResultSet rst3 = statement.executeQuery("SELECT * FROM HH ORDER BY RATING DESC");
            String hhdetail= "";
            String hhid="";
            String hhdomain="";
            String rating ="";
    %>
    <table border="1" bgcolor="#C0C0C0">
            <tr bgcolor="#808080">
              <td align="center">
          <b>Headhunter Name </b>
        </td>
        <td align="center">
          <b>Headhunter Id </b>
        </td>
        <td align="center">
          <b>Domain </b>
        </td>
        <td align="center">
          <b>Rating </b>
        </td>

      </tr>
    <%
            while(rst3.next())
            {
                hhdetail = rst3.getString("HHNAME");
                hhid = rst3.getString("HHID");
                hhdomain = rst3.getString("DOMAIN");
                rating = rst3.getString("RATING");
    %>
    <tr>
            <td>
            <B><%=hhdetail%></B>
            </td>
            <td>
            <B><%=hhid%></B>
            </td>
            <td>
            <B><%=hhdomain%></B>
            </td>
            <td>
            <B><%=rating%></B>
            </td>

    </tr>

    <%
            } //while
    %>
    </table>

    <%
            ResultSet rst4 = statement.executeQuery("SELECT COUNT(*) as ARNO FROM AR");
            int arno= 0;
            String arname = "No of Active Registries Active";

            while(rst4.next())
            {
                arno = rst4.getInt("ARNO");
            }
    %>
   <table>
    <tr>
            <td height = "28">
            <B><%=arname%> </B>
            </td>

            <td height = "28">
            <% if (arno != 0 )
            { %>
            <B><%=arno%></B>

            <% }
            else
            { %>
            <B> No Active ARs </B>
            <% } %>
            </td>
    </tr>
    <br>
    <tr>
            <td>
            <b>
            <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor_ar_details.jsp">
            Display Details</a>
            </b>
            </td>
    </tr>

   <tr>
            <td height = "28">
            <B> No. of Queries being processed: </B>
            </td>
            <td height = "28">
            <B> <%=j%> </B>
            </td>
   </tr>
<%
            statement.close();
            dbconn.close();
%>
</table>
</body>
</html>
```

Monitor_ar_details.jsp

```
/**
 * This JSP page collects the details of all the ARs.
 *
 * @author Srikanth Reddy
 * @date Feb 2005
 */
<%@ page language="java" import="java.sql.*, RMIHelper" %>
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
            RMIHelper obj2 = new RMIHelper();
            int j = -1;
            j = obj2.queSize();
%>

<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #F8F7D9>

<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            Statement statement = dbconn.createStatement();
%>
  <table width="550" border="0" cellspacing="1" cellpadding="0" height="84">
    <tr>
            <td bgcolor="#800000" height="30" colspan="5" align="center" bordercolor="#C0C0C0" >
            <font color="#FFFFFF" size="5">
            <b> Global Snapshot of URDS </b>
            </font>
            </td>
    </tr>
    <%
            ResultSet rst = statement.executeQuery("select * from dsm");
            String dsmval="";
            String dsmname = "Domain Security Manager";
```

```
        while(rst.next())
        {
            dsmval=rst.getString("DSMID");
        }
%>
<tr>
        <td height="28">
        <B><%=dsmname%> </B>
        </td>

        <td height="28">
        <% if (dsmval != "" )
        { %>

        <B><%=dsmval%></B>

        <% }
        else
        { %>

        <B> No Active DSM </B>

        <% } %>
        </td>
</tr>

<br>
<br>

<%
        ResultSet rst1 = statement.executeQuery("select * from qm");
        String qmval="";
        String qmname = "Query Manager";

        while(rst1.next())
        {
            qmval=rst.getString("QMID");
        }
%>
<tr>
        <td height = "28">
        <B><%=qmname%> </B>
        </td>

        <td height = "28">
        <% if (qmval != "" )
        { %>
        <B><%=qmval%></B>

        <% }
        else
        { %>
        <B> No Active QM </B>
        <% } %>
        </td>
</tr>

<%
        ResultSet rst2 = statement.executeQuery("SELECT COUNT(*) as HHNO FROM HH");
        int hhno= 0;
        String hhname = "No of Headhunters Active";

        while(rst2.next())
        {
            hhno = rst2.getInt("HHNO");
        }
   %>

   <tr>
        <td height = "28">
        <B><%=hhname%> </B>
        </td>

        <td height = "28">
        <% if (hhno != 0 )
        { %>
        <B><%=hhno%></B>

        <% }
        else
        { %>
        <B> No Active HHs </B>
        <% } %>
        </td>
   </tr>
   <tr>
        <td>
        <b>
        <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor_hh_details.jsp">
        Display Details</a>
        </b>
        </td>
  </tr>

   <%
        ResultSet rst4 = statement.executeQuery("SELECT COUNT(*) as ARNO FROM AR");
        int arno= 0;
        String arname = "No of Active Registries Active";

        while(rst4.next())
        {
            arno = rst4.getInt("ARNO");
        }
   %>

   <tr>
        <td height = "28">
        <B><%=arname%> </B>
        </td>

        <td height = "28">
        <% if (arno != 0 )
        { %>
        <B><%=arno%></B>

        <% }
        else
        { %>
        <B> No Active ARs </B>
        <% } %>
        </td>
```

```
    </tr>
   <tr>
            <td width="324">
            <b>
            <a href="http://magellan.cs.iupui.edu:8080/srikanth/jsp/Monitor.jsp">
            Hide Details</a>
            </b>
            </td>
   </tr>
   </table>
    <%
            ResultSet rst5 = statement.executeQuery("SELECT * FROM AR");
            String ardetail= "";
            String arid="";
            String ardomain = "";
    %>

    <table border="1" bgcolor="#C0C0C0">
      <tr bgcolor="#808080">
       <td align="center">
         <b>Active Registry Name </b>
       </td>
       <td align="center">
         <b>Active Registry Id </b>
       </td>
       <td>
         <b>Domain </b>
       </td>
      </tr>


    <%
            while(rst5.next())
            {
               ardetail = rst5.getString("ARNAME");
               arid = rst5.getString("ARID");
               ardomain = rst5.getString("DOMAIN");
    %>
   <tr>
   <td>
            <B><%=ardetail%></B>
   </td>

   <td>
       <B><%=arid%></B>
   </td>

   <td>
            <B><%=ardomain%></B>
   </td>
   </tr>

<%
            } //while
%>
</table>
<table>
   <tr>
            <td height = "28">
            <B> No. of Queries being processed: </B>
            </td>
            <td height = "28">
            <B> <%=j%> </B>
            </td>
   </tr>
<%
            statement.close();
            dbconn.close();
%>
</table>
</body>
</html>
```

HHUtilization.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<form  name="utilHH" method="post" action="processHHUtil.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
   <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <b>Utilization of Headhunter</b></p>
</tr>

<tr valign="center">

     <td height="21" colspan="2"><b>Select Headhunter's Location</b></td>

     <td valign="top" colspan="4" height="21" width="478">
     <select name="hhLocation">

<%
    try
    {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            System.out.println("Database connected");
            Statement statement = dbconn.createStatement();
            ResultSet rst = statement.executeQuery("select hhid from hh");
            System.out.println("Query executed");
            String hhlist="";

            while(rst.next())
            {
                hhlist=rst.getString("HHID");
%>
          <option value="<%= hhlist %>" > <%= hhlist %></option>
<%
            } //while
            statement.close();
            dbconn.close();
    }catch(Exception e)
    {
            System.out.println(e.getMessage());
    }
```

```
%>

     </select></td>
</tr>

</table>
<br>
             <p align="center">
             <input type="submit" name="HHUtil" value="Get HH Utilization">
</p>
</form>

</body>
</html>
```

procssHHutil.jsp

```
<%@ page language="java" %>
<%@ page import="java.util.*, java.lang.*, java.text.*, RMIHelper, LineGraphTest1"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
   RMIHelper obj1 = new RMIHelper();
   LineGraphTest1 obj2 = new LineGraphTest1();
   ArrayList utilPerList = new ArrayList();
   java.text.DecimalFormat df = new java.text.DecimalFormat("#0.00");
%>
<%
 if(request.getParameter("HHUtil")!=null)
 {
   String hhAddress = (String) request.getParameter("hhLocation");
   if(hhAddress != null)
   {
             try
             {
                          utilPerList = obj1.HHUtil(hhAddress);
                          System.out.println("before drawGraph()");
                          obj2.main(utilPerList, hhAddress);
                          Thread.sleep(2000);
             }
             catch(Exception e)
             {
                          System.out.println("error is "+e.getMessage());
                          e.printStackTrace();
             }
   }
   else
   {
%>
             <p> Invalid Headhunter Location </p>
<%
   }
 }
 else
 {
%>
   <p> Sorry, unable to determine at this time </p>
<%
 }
%>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Headhunter Untilization</title>
</head>

<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>

 <b> Headhunter's Utilization Graph </b>
<br>
<img src="LineGraphTest.png">

</center>
</body>
</html>
```

QMUtilization.jsp

```
<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #FFFFFF>
<form  name="utilQM" method = "post" action="processQMUtil.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
   <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <B> Utilization of Query Manager </B>
</tr>
</table>
<p align="center">
            <br>
            <input type="submit" name="QMUtil" value="Get QM Utilization">
</p>
</form>

</body>
</html>
```

processQMutil.jsp

```
<%@ page language="java" %>
<%@ page import="java.util.*, java.lang.*, java.text.*, RMIHelper, LineGraphTest2"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
   RMIHelper obj1 = new RMIHelper();
   LineGraphTest2 obj2 = new LineGraphTest2();
   ArrayList utilPerList = new ArrayList();
   java.text.DecimalFormat df = new java.text.DecimalFormat("#0.00");
%>
<%
   try
   {
            String qmAddress = "//magellan.cs.iupui.edu:2000/QueryManager";
      utilPerList = obj1.QMUtil();
```

```
                obj2.main(utilPerList, qmAddress);
                Thread.sleep(2000);
    }
    catch(Exception e)
    {
                System.out.println("Exception in processQMutil "+e.getMessage());
    }
%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Query Manager Untilization</title>
</head>

<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
 <b> Query Manager's Utilization Graph </b>
</font>
<br>
<img src="LineGraphTest1.png">
</center>
</body>
</html>
```

getTraffic.jsp

```
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #F8F7D9>
  <table width="550" border="0" cellspacing="1" cellpadding="0">
     <tr>
                <td bgcolor="#800000" height="30" colspan="5" align="center" bordercolor="#C0C0C0" >
                <font color="#FFFFFF" size="5">
                <B> Get Message Traffic </B>
                </font>
                                </td>
    </tr>
   </table>

<form  name="Traffic" method = "post" action="processgetTraffic.jsp" >
<p align="center">
                <br>
                <br>
                <br>

                <input type="submit" name="getTraffic" value="Get Message Traffic">
</p>
</form>
</body>
</html>
```

processgetTraffic.jsp

```
<%@ page language="java" %>
<%@ page import="java.util.*, java.lang.*, java.io.*, RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
<%
RMIHelper obj1 = new RMIHelper();
ArrayList list3 = new ArrayList();

Integer a1 = null;
Integer b1 = null;
Integer c1 = null;
int a=0, b=0, c=0;
%>

<%
try
{
        list3 = obj1.msgTraffic();
        if (list3!=null && list3.size() ==3)
        {
                a1 = (Integer) list3.get(0);
                a = a1.intValue();
                b1 = (Integer) list3.get(1);
                b = b1.intValue();
                c1 = (Integer) list3.get(2);
                c = c1.intValue();
        }
        else
        {
                System.out.println("List is null");
        }
}
catch(Exception ex)
{
        System.out.println("Excp"+ex.getMessage());
}
%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Message Traffic</title>
</head>

<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
if(list3!=null && list3.size()==3)
{
%>

<table border="1" bgcolor="#C0C0C0" width="550">
   <tr bgcolor="#808080">
     <td>
        <b> Classification </b>
     </td>
     <td>
```

```
                <b> No of Messages</b>
      </td>
    <tr>
                <td height = "28" width="206">
                <B>Authentication Messages</B>
                </td>

                <td height = "28" width="349">
                <B><%=a%></B>
                </td>
   </tr>
   <tr>
                <td height = "28" width="206">
                <B>Query Processing Messages</B>
                </td>

                <td height = "28" width="349">
                <B><%=b%></B>
                </td>
   </tr>
   <tr>
                <td height = "28" width="206">
                <B>Component Update</B>
                </td>

                <td height = "28" width="349">
                <B><%=c%></B>
                </td>
   </tr>
</table>

<%
   }
   else
   {
%>
  <p> Sorry, unable to determine at this time </p>
<%
   }
%>
</font>
</center>
</body>
</html>
```

status.jsp

```
<%@ page language="java" import="java.sql.*, RMIHelper" %>
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
                RMIHelper obj2 = new RMIHelper();
                int j = 0;
                j = obj2.queSize();
%>

<html>
<head>
<META HTTP-EQUIV="Refresh" CONTENT="10; URL=http://magellan.cs.iupui.edu:8080/srikanth/jsp/status.jsp">
</head>
```

```
<body bgcolor = #FFFFFF>

<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            Statement statement = dbconn.createStatement();
%>
  <table width="190" border="0" cellspacing="1" cellpadding="0">
    <tr>
            <td bgcolor="#800000" height="20" colspan="2" align="center" bordercolor="#C0C0C0" >
            <font color="#FFFFFF" size="2">
            <b> URDS Snapshot</b>
            </font>
            </td>
    </tr>
    <%
            ResultSet rst = statement.executeQuery("select COUNT(*) as DSMNO from dsm");
            int dsmval=0;
            String dsmname = "DSM:";
    %>
      <tr>
            <td height="20">
            <B><%=dsmname%></B>
            </td>
    <%
            while(rst.next())
            {
                dsmval=rst.getInt("DSMNO");
            }
    %>
            <td height="20">
            <% if (dsmval != 0 )
            { %>

            <B><%=dsmval%></B>

            <% }
            else
            { %>
            <B> No Active DSM </B>
            <% } %>
            </td>
    </tr>

    <%
            ResultSet rst1 = statement.executeQuery("select COUNT(*) as QMNO from qm");
            int qmval=0;
            String qmname = "QM";

      while(rst1.next())
            {
                //qmname=rst.getString("QMNAME");
                qmval=rst.getInt("QMNO");
            }
     %>
    <tr>
      <td height = "20">
      <B><%=qmname%> </B>
      </td>
```

```
            <td height = "20">
            <% if (qmval != 0 )
            { %>
            <B><%=qmval%></B>

            <% }
            else
            { %>
            <B> No Active QM </B>
            <% } %>
            </td>
     </tr>

     <%
            ResultSet rst2 = statement.executeQuery("SELECT COUNT(*) as HHNO FROM HH");
            int hhno= 0;
            String hhname = "No of HHs:";

            while(rst2.next())
            {
                 hhno = rst2.getInt("HHNO");
            }
     %>

    <tr>
            <td height = "20">
            <B><%=hhname%> </B>
            </td>

            <td height = "20">
            <% if (hhno != 0 )
            { %>
            <B><%=hhno%></B>

            <% }
            else
            { %>
            <B> No Active HHs </B>
            <% } %>
            </td>
    </tr>

    <%
            ResultSet rst3 = statement.executeQuery("SELECT COUNT(*) as ARNO FROM AR");
            int arno= 0;
            String arname = "No of ARs:";

            while(rst3.next())
            {
                 arno = rst3.getInt("ARNO");
            }
    %>

    <tr>
            <td height = "20">
            <B><%=arname%> </B>
            </td>

            <td height = "20">
            <% if (arno != 0 )
            { %>
            <B><%=arno%></B>

            <% }
            else
            { %>
            <B> No Active ARs </B>
            <% } %>
            </td>
   </tr>

   <tr>
            <td height = "20">
            <B>No. of Queries: </B>
            </td>
            <td height = "20">
            <B> <%=j%> </B>
            </td>
   </tr>
<%
            statement.close();
            dbconn.close();
%>
</table>
</body>
</html>
```

errorsDisplay.jsp

```
<%@ page language="java" import="java.sql.*, RMIHelper" %>
<%@ page import="java.util.*" %>

<html>

<head>
<META HTTP-EQUIV="Refresh" CONTENT="10;
URL=http://magellan.cs.iupui.edu:8080/srikanth/jsp/errorsDisplay.jsp">
<title>Error Page</title>
</head>

<body>
  <table width="190" border="0" cellspacing="1" cellpadding="0">
    <tr>
            <td bgcolor="#800000" height="20" colspan="2" align="center" bordercolor="#C0C0C0" >
            <font color="#FFFFFF" size="2">
            <b> Errors </b>
            </font>
            </td>
    </tr>

    <tr>
    <%
            RMIHelper obj2 = new RMIHelper();
            StringBuffer errBuff = (StringBuffer) obj2.getError();

            if(errBuff.length() == 0)
            {
              System.out.println("No Errors so far");
    %>
            <td>
                        <p align="left">No Errors so far </p>
            </td>
    <%
            }
            else
            {
              System.out.println("Errors encountered \n");
              System.out.println("printing StringBuffer "+ errBuff.toString());
    %>
              <td>
              <p align="left"><b><font color="#FF0000">Error encountered </font> </b>
              <br>
              <%=errBuff.toString()%>
              </p>
              </td>
    <%
            }
    %>
    </tr>

</table>
</body>
</html>
```

ExceptionHandler.jsp

```
<%@ page isErrorPage="true" import="java.io.*" %>
<html>
<head>
            <title>Exceptional Even Occurred!</title>
            <style>
            body, p { font-family:Tahoma; font-size:10pt; padding-left:30; }
            pre { font-size:8pt; }
            </style>
</head>
<body>

<%-- Exception Handler --%>
<font color="red">
<%= exception.toString() %><br>
</font>

<%
out.println("<!--");
StringWriter sw = new StringWriter();
PrintWriter pw = new PrintWriter(sw);
exception.printStackTrace(pw);
out.print(sw);
sw.close();
pw.close();
out.println("-->");
%>

</body>
</html>
```

mainSI.jsp

```
<%
if(request.getParameter("mainSI.jsp") != null)
   {
      if(request.getParameter("selection").trim().equals("SubmitQuery"))
      {
```

```
%>
        <jsp:forward page="newQuery.jsp" />
<%
        }
        else if(request.getParameter("selection").trim().equals("DisplayResults"))
        {
%>
        <jsp:forward page="displayResults.jsp" />
<%
        }
     }
%>

<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<form method="post" action="mainSI.jsp">
<br>
<p>
<table border="1" bgcolor="#C0C0C0">
<br>
<tr bgcolor="#808080" align="center">
  <td> Select </td>
  <td> Task </td>
</tr>
<tr>
  <td align="center">
     <input type="radio" name="selection" value="SubmitQuery" checked>
  </td>
  <td>
     Submit a query for processing
  </td>
</tr>
<tr>
  <td align="center">
     <input type="radio" name="selection" value="DisplayResults">
  </td>
  <td>
     Display results for a query
  </td>
</tr>
</table>
<br>
<br>
  <input type="submit" name="mainSI.jsp" value="Submit">
</form>
<br>
<br>
<br>
</center>
</body>
</html>
```

newQuery.jsp

```
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
```

```
<%@ page import="java.util.*" %>
<body bgcolor = #F8F7D9>
<form name="newQuery" method="post" action="processQuery.jsp">

 <%
               Integer localCounter=(Integer)session.getAttribute("queryId");
               int myCounter=localCounter.intValue();
               myCounter = myCounter + 1;
               session.setAttribute("queryId",(Object) (new Integer(myCounter)));

%>
  <table width="600" border="0" cellspacing="1" cellpadding="0">
    <tr>
    <td bgcolor="#CCCCCC" valign="top" height="30" colspan="5" >
               <p align="center">
               <b>System Integrator View of URDS </b>
               </td>
    </tr>
    </p>
    <tr>
     <td height="30" valign="top" colspan="2"><b>Query ID</b></td>
      <td colspan="3" valign="top" height="30">
        <%=myCounter%>
      </td>
     </tr>

     <tr>
     <td height="30" valign="top" colspan="2"><b>Component Name</b></td>

      <td colspan="3" valign="top" height="30">
        <input type="text" name="compName" size="30" value="Document">
      </td>
     </tr>

     <tr>
      <td height="30" valign="top" colspan="2"><b>Response Time</b> </td>

      <td colspan="3" valign="top" height="30">
        <input type="text" name="respTime" size="10" value="12000">
      </td>
     </tr>
    </table>
    <table>
               <tr>
     <td>
         <input type="submit" name="newQuery" value="Submit Query" style="float: left">
     </td>
     <td>
         <input type="reset" name="Submit2" value="Reset Form" style="float: left">
     </td>
    </tr>
    </table>
  </form>
 </body>
 </html>
```

processQuery.jsp

```
<%@ page import="java.util.*, java.lang.*, RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
```

```
<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("newQuery")!=null)
 {
    String Qid = (String) request.getParameter("queryID");
    String cName = (String) request.getParameter("compName");
    String temp = (String) request.getParameter("respTime");
    long reTime = Long.parseLong(temp);
    if(cName != null && reTime != 0)
    {
            status = (int)Helper.subQuery(cName, reTime);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
%>
 <p>  Query submitted successfully   </p>
<%
   }
   else
   {
%>
 <p>  Invalid parameters: resubmit the query   </p>
<%
   }
%>
</center>
</body>
</html>
```

displayResults.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="displayResults" method = "post" action="Results.jsp" >
<table width="774" border="0" cellspacing="1" cellpadding="0">
```

```
<tr>
   <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <b> Get Query Results</b></p>
</tr>

<tr valign="center">

    <td height="21" colspan="2"><b>Select Query ID</b></td>

    <td valign="top" colspan="4" height="21" width="478">
    <select name="queryID">

<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            Statement statement = dbconn.createStatement();
            ResultSet rst = statement.executeQuery("select DISTINCT(QID) from qmresults");
            String qidlist="";

            while(rst.next())
            {
                qidlist=rst.getString("QID");
%>
     <option value="<%= qidlist %>" > <%= qidlist %></option>
<%
            } //while
            statement.close();
            dbconn.close();
%>

    </select></td>
</tr>
</table>

<p align="center">
            <input type="submit" name="getResults" value="Get Results">
            <input type="reset" name="Submit2" value="Reset Form">
</p>

</form>
</body>
</html>
```

Results.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page import="java.util.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>

<html>
<head>
<meta http-equiv="PRAGMA" content="NO-CACHE">
</head>
<body bgcolor = #F8F7D9>
<%
  String Qid = "";
  if(request.getParameter("getResults") != null)
    {
            Qid = (String) request.getParameter("queryID").trim();
      }
    else
      {
%>
      <p> Invalid Entry </p>
<%
      }
%>

<%
   Class.forName("oracle.jdbc.driver.OracleDriver");
   Connection dbconn = DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl",
"urdsmon", "reddi");
   Statement statement = dbconn.createStatement();
%>
 <table width="700" border="0" cellspacing="1" cellpadding="0" height="84">
   <tr>
            <td bgcolor="#800000" height="30" colspan="5" align="center" bordercolor="#C0C0C0" >
            <font color="#FFFFFF" size="5">
            <b> Query Results </b>
            </font>
            </td>
   </tr>

   <tr>
    <td width="246">
            <p align="left">
            <b> <font size="4">Details of the Submitted Query </font> </b>
    </td>
   </tr>
   <tr>
    <td width="246">
      <b> Query ID </b>
    </td>

    <td width="451">
      <b> <%=Qid%> </b>
    </td>
   </tr>
  </table>
  <hr>
  <table>

   <tr>
    <td>
      <b> <font size="4">Results </font> </b>
    </td>
   </tr>
   <%
            String querystring = "SELECT * FROM QMRESULTS WHERE QID = '"+Qid+"'";
            ResultSet rst = statement.executeQuery(querystring);

            String queryID = "";
            float respTime = 0;
            int cno = 0;

            while(rst.next())
            {
                queryID = rst.getString("QID");
                respTime = rst.getFloat("RTIME");
                cno = rst.getInt("CMATCH");
            }
    %>

    <tr>
            <td height="28" width="207">
            <B> No of Components Matched </B>
            </td>

            <td height="28" width="490">
            <B><%=cno%></B>
    </tr>
    <br>
    <tr>
            <td height="28" width="207">
            <B> Response Time of the Query </B>
            </td>

            <td height="28" width="490">
            <B><%=respTime%></B>
    </tr>
   </table>
   <table border="1" bgcolor="#C0C0C0" width="704">
   <tr bgcolor="#808080">
     <td>
            <b> Component Name </b>
     </td>
     <td>
            <b> Component ID </b>
     </td>
     <td><b> Component Rating </b></td>
    <%
            ResultSet rst1 = statement.executeQuery("select cname, cid, crating from urdsresults where qid
='"+queryID+"' order by crating desc");
            String CoID="";
            String CoName="";
            float CoRating = 0;

            while(rst1.next())
            {
                CoName=rst1.getString("CNAME");
                CoID=rst1.getString("CID");
                CoRating = rst1.getFloat("CRATING");
    %>
    <tr>
            <td height = "28" width="206">
            <B><%=CoName%> </B>
            </td>

            <td height = "28" width="349">
            <B><%=CoID%></B>
            </td>

            <td>
            <B> <%=CoRating%> </B>
            </td>
   </tr>
<%
            }
            statement.close();
            dbconn.close();
```

```
%>
</table>
</body>
</html>
```

DSMStart.jsp

```
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
   <tr>
      <td height="50" valign="middle" bgcolor="#999965">
      <h1><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h1>
      </td>
   </tr>
</table>

<form  name="startDSM" method = "post" action="processDSM.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
   <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <b> Starting Domain Security Manager </b> </p>
</tr>
<tr>
     <td height="25" valign="center" colspan="3" ><b>Enter the location</b> <br>
                        <font size="2">(where DSM has to be started)</font></td>

     <td colspan="3" valign="center" height="25" >
       <input type="text" name="dsmLocation" size="30">
     </td>
</tr>

<tr>

     <td height="25" valign="center" colspan="3" ><b>Enter the port no <br> </b></td>

     <td colspan="3" valign="center" height="25" >
       <input type="text" name="portNo" size="10">
     </td>
</tr>

</table>

<p align="center">
            <input type="submit" name="StartDSM" value="Start DSM">
            <input type="reset" name="Submit2" value="Reset Form">
</p>
</form>
</body>
</html>
```

processDSM.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
```

```
<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StartDSM")!=null)
 {
    String dsmAddress = (String) request.getParameter("dsmLocation");
    String port = request.getParameter("portNo");
    Integer portno =  new Integer(Integer.parseInt(port));
    if(dsmAddress != null && portno.intValue() >1000)
    {
            status = (int)Helper.startDSM(dsmAddress, portno.intValue());
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   DSM started successfully
<%
   }
   else
   {
%>
   Invalid parameters: DSM not started
<%
   }
%>
</center>
</body>
</html>
```

HHStart.jsp

```
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<%@ page language="java" import="java.sql.*" %>
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System 
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="startHH" method = "post" action="processHH.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
```

```
                <p align="center">
                <b> Starting Headhunter </b> </p>
</tr>
<tr>

      <td height="25" valign="center" colspan="2"><b>Enter the location</b> <br>
                              <font size="2">(where HH has to be started)</font></td>

      <td colspan="3" valign="center" height="25">
        <input type="text" name="hhLocation" size="50">
      </td>
</tr>

<tr>
      <td height="21" valign="center" colspan="2"><b>Enter the port no </b></td>

      <td colspan="3" valign="center" height="21">
        <input type="text" name="portNo" size="10">
      </td>
</tr>
<tr valign="center">
      <td height="21" colspan="2"><b>Select DSM Location</b></td>
      <td valign="top" colspan="4" height="21">
      <select name="dsmLocation">
<%
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                System.out.println("Database connected");
                Statement statement = dbconn.createStatement();
                ResultSet rst = statement.executeQuery("select dsmid from dsm");
                System.out.println("Query executed");
                String dsmlist="";
                while(rst.next())
                {
                     dsmlist=rst.getString("DSMID");
%>
        <option value="<%= dsmlist %>" > <%= dsmlist %></option>
<%
                } //while
                statement.close();
                dbconn.close();
%>
      </select>
    </td>
</tr>
<tr valign="top">
      <td height="21" colspan="2"><b>Domain</b></td>
      <td valign="top" colspan="4" height="21">
        <select name="domain">
                              <option value="Document" selected>Document</option>
                              <option value="Banking" >Banking</option>
                              <option value="Finance" >Finance</option>
                              <option value="Manufacturing">Manufacturing</option>
        </select>
    </td>
</tr>
<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Headhunter Name </b></td>
      <td colspan="3" height="25">
        <input type="text" name="hhName" size="25">
      </td>
</tr>

<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Headhunter Password </b></td>
      <td colspan="3" valign="center" height="25">
        <input type="password" name="hhPwd" size="25">
      </td>
</tr>
</table>
<p align="center">
                <input type="submit" name="StartHH" value="Start HH">
                <input type="reset" name="Submit1" value="Reset Form">
</p>
</form>
</body>
</html>
```

processHH.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StartHH")!=null)
 {
    String hhAddress = (String) request.getParameter("hhLocation");
    String port = request.getParameter("portNo");
    String dsmLocation = (String) request.getParameter("dsmLocation");
    String domain = (String) request.getParameter("domain");
    String name = (String) request.getParameter("hhName");
    String pwd = (String) request.getParameter("hhPwd");

    Integer portno =  new Integer(Integer.parseInt(port));
    if(hhAddress != null && dsmLocation != null && domain != null && portno.intValue() >1000 && name != null
&& pwd != null)
    {
      status = (int)Helper.startHH(hhAddress, portno.intValue(), dsmLocation, domain, name, pwd);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   HH started successfully
<%
   }
   else
   {
%>
```

```
   Invalid parameters: HH not started
<%
   }
%>
</center>
</body>
</html>
```

ARStart.jsp

```
<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<%@ page language="java" import="java.sql.*" %>

<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="startAR" method = "post" action="processAR.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">
<tr>
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
                <p align="center">
                <b> Starting Active Registry </b> </p>
</tr>
<tr>
      <td height="25" valign="center" colspan="2"><b>Enter the location</b> <br>
                <font size="2">(where AR has to be started)</font></td>

      <td colspan="3" valign="center" height="25">
        <input type="text" name="arLocation" size="50">
      </td>
</tr>

<tr>
      <td height="21" valign="center" colspan="2"><b>Enter the port no </b></td>
      <td colspan="3" valign="center" height="21">
        <input type="text" name="portNo" size="10">
      </td>
</tr>

<tr>
      <td height="21" valign="center" colspan="2"><b>Enter the port no where Component is running </b></td>
      <td colspan="3" valign="center" height="21">
        <input type="text" name="coportNo" size="10">
      </td>
</tr>

<tr valign="center">
      <td height="21" colspan="2"><b>Select DSM Location</b></td>
      <td valign="top" colspan="4" height="21">
      <select name="dsmLocation">
<%
```

```
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                System.out.println("Database connected");
                Statement statement = dbconn.createStatement();
                ResultSet rst = statement.executeQuery("select dsmid from dsm");
                System.out.println("Query executed");
                String dsmlist="";
                while(rst.next())
                {
                    dsmlist=rst.getString("DSMID");
%>
        <option value="<%= dsmlist %>" > <%= dsmlist %></option>
<%
                } //while
                statement.close();
                dbconn.close();
%>
      </select>
    </td>
</tr>
<tr valign="top">
      <td height="21" colspan="2"><b>Domain</b></td>
      <td valign="top" colspan="4" height="21">
        <select name="domain">
                        <option value="Document" selected>Document</option>
                        <option value="Banking" >Banking</option>
                        <option value="Finance" >Finance</option>
                        <option value="Manufacturing">Manufacturing</option>
        </select>
    </td>
</tr>

<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Active Registry's Name </b></td>

      <td colspan="3" height="25">
        <input type="text" name="arName" size="25">
      </td>
</tr>

<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Active Registry's Password </b></td>
      <td colspan="3" valign="center" height="25">
        <input type="password" name="arPwd" size="25">
      </td>
</tr>
</table>

<p align="center">
                <input type="submit" name="StartAR" value="Start AR">
                <input type="reset" name="Submit1" value="Reset Form">
</p>
</form>
</body>
</html>
```

processAR.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
```

```
<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StartAR")!=null)
 {
    String arAddress = (String) request.getParameter("arLocation");
    String port = request.getParameter("portNo");
    String coport = request.getParameter("coportNo");
    String dsmLocation = (String) request.getParameter("dsmLocation");
    String domain = (String) request.getParameter("domain");
    String name = (String) request.getParameter("arName");
    String pwd = (String) request.getParameter("arPwd");
    Integer portno =  new Integer(Integer.parseInt(port));
    Integer coportno =  new Integer(Integer.parseInt(coport));

    if(arAddress != null && dsmLocation != null && domain != null && portno.intValue() >1000 &&
coportno.intValue() >1000 && name != null && pwd != null)
    {
      status = (int)Helper.startAR(arAddress, portno.intValue(), coportno.intValue(), dsmLocation, domain, name, pwd);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   AR started successfully
<%
   }
   else
   {
%>
   Invalid parameters: AR not started
<%
   }
%>
</center>
</body>
</html>
```

QMStart.jsp

```
<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<%@ page language="java" import="java.sql.*" %>
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
```

```
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>
<form  name="startQM" method = "post" action="processQM.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
                <p align="center">
                <b> Starting Query Manager </b> </p>
</tr>
<tr>
      <td height="25" valign="center" colspan="2"><b>Enter the location</b> <br>
                <font size="2">(where QM has to be started)</font></td>

      <td colspan="3" valign="center" height="25">
        <input type="text" name="qmLocation" size="50">
      </td>
</tr>

<tr>
      <td height="21" valign="center" colspan="2"><b>Enter the port no </b></td>
      <td colspan="3" valign="center" height="21">
        <input type="text" name="portNo" size="10">
      </td>
</tr>

<tr valign="center">

      <td height="21" colspan="2"><b>Select DSM Location</b></td>
      <td valign="top" colspan="4" height="21">
      <select name="dsmLocation">
<%
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                System.out.println("Database connected");
                Statement statement = dbconn.createStatement();
                ResultSet rst = statement.executeQuery("select dsmid from dsm");
                System.out.println("Query executed");
                String dsmlist="";
                while(rst.next())
                {
                     dsmlist=rst.getString("DSMID");
%>
        <option value="<%= dsmlist %>" > <%= dsmlist %></option>
<%
                } //while
                statement.close();
                dbconn.close();
%>
      </select>
    </td>
</tr>
<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Query Manager's Name </b></td>
      <td colspan="3" height="25">
        <input type="text" name="qmName" size="25">
      </td>
</tr>
```

```
<tr>
      <td height="25" valign="center" colspan="2"><b>Enter Query Manager's Password </b></td>
      <td colspan="3" valign="center" height="25">
        <input type="password" name="qmPwd" size="25">
      </td>
</tr>
</table>
<p align="center">
                <input type="submit" name="StartQM" value="Start QM">
                <input type="reset" name="Submit1" value="Reset Form">
</p>
</form>
</body>
</html>
```

processQM.jsp

```
<%@ page import="java.util.*, java.lang.*, RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StartQM")!=null)
 {
    String qmAddress = (String) request.getParameter("qmLocation");
    String port = request.getParameter("portNo");
    String dsmLocation = (String) request.getParameter("dsmLocation");
    String name = (String) request.getParameter("qmName");
    String pwd = (String) request.getParameter("qmPwd");

    Integer portno =  new Integer(Integer.parseInt(port));
    if(qmAddress != null && dsmLocation != null && portno.intValue() >1000 && name != null && pwd != null)
    {
      status = (int)Helper.startQM(qmAddress, portno.intValue(), dsmLocation, name, pwd);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   Query Manager started successfully
<%
   }
   else
   {
%>
   Invalid parameters: Query Manager not started
<%
   }
%>
```

```
</center>
</body>
</html>
```

processQM.jsp

```
<%@ page import="java.util.*, java.lang.*, RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StartQM")!=null)
 {
    String qmAddress = (String) request.getParameter("qmLocation");
    String port = request.getParameter("portNo");
    String dsmLocation = (String) request.getParameter("dsmLocation");
    String name = (String) request.getParameter("qmName");
    String pwd = (String) request.getParameter("qmPwd");

    Integer portno =  new Integer(Integer.parseInt(port));
    if(qmAddress != null && dsmLocation != null && portno.intValue() >1000 && name != null && pwd != null)
    {
      status = (int)Helper.startQM(qmAddress, portno.intValue(), dsmLocation, name, pwd);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   Query Manager started successfully
<%
   }
   else
   {
%>
   Invalid parameters: Query Manager not started
<%
   }
%>
</center>
</body>
</html>
```

DSMStop.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
```

```
<table border ="0" cellpadding="0" cellspacing="0">
   <tr>
      <td height="50" valign="middle" bgcolor="#999965">
      <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
      </td>
   </tr>
</table>

<form  name="stopDSM" method = "post" action="killDSM.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
   <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <b> Stop a DSM </b> </p>
</tr>

<tr valign="center">
     <td height="21" colspan="2"><b>Select DSM Location</b></td>
     <td valign="top" colspan="4" height="21" width="478">
     <select name="dsmLocation">
<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            System.out.println("Database connected");
            Statement statement = dbconn.createStatement();
            ResultSet rst = statement.executeQuery("select dsmid from dsm");
            System.out.println("Query executed");
            String dsmlist="";
            while(rst.next())
            {
                dsmlist=rst.getString("DSMID");
%>
      <option value="<%= dsmlist %>" > <%= dsmlist %></option>
<%
            } //while
            statement.close();
            dbconn.close();
%>

     </select>
   </td>
</tr>

</table>

<p align="center">
            <input type="submit" name="StopDSM" value="Stop DSM">
            <input type="reset" name="Submit2" value="Reset Form">
</p>

</form>
</body>
</html>
```

killDSM.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
```

```
<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StopDSM")!=null)
 {
    String dsmAddress = (String) request.getParameter("dsmLocation");
    if(dsmAddress != null)
    {
      status = (int)Helper.stopDSM(dsmAddress);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   DSM stopped successfully
<%
   }
   else
   {
%>
   Invalid parameters: DSM not stopped
<%
   }
%>
</center>
</body>
</html>
```

HHStop.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
   <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="stopHH" method = "post" action="killHH.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
```

```
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
            <p align="center">
            <b> Stop Headhunter</b></p>
</tr>

<tr valign="center">

     <td height="21" colspan="2"><b>Select Headhunter's Location</b></td>

     <td valign="top" colspan="4" height="21" width="478">
     <select name="hhLocation">

<%
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
            System.out.println("Database connected");
            Statement statement = dbconn.createStatement();
            ResultSet rst = statement.executeQuery("select hhid from hh");
            System.out.println("Query executed");
            String hhlist="";

            while(rst.next())
            {
                hhlist=rst.getString("HHID");
%>
      <option value="<%= hhlist %>" > <%= hhlist %></option>
<%
            } //while
            statement.close();
            dbconn.close();
%>

     </select></td>
</tr>


</table>

<p align="center">
            <input type="submit" name="StopHH" value="Stop HH">
            <input type="reset" name="Submit2" value="Reset Form">
</p>

</form>
 </body>
 </html>
```

killHH.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StopHH")!=null)
 {
    String hhAddress = (String) request.getParameter("hhLocation");
    if(hhAddress != null)
    {
      status = (int)Helper.stopHH(hhAddress);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   Headhunter stopped successfull
<%
   }
   else
   {
%>
   Invalid parameters: Headhunter not stopped
<%
   }
%>
</center>
</body>
</html>
```

ARStop.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>

<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="stopAR" method = "post" action="killAR.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
                <p align="center">
                <b> Stop ActiveRegistry</b></p>
</tr>

<tr valign="center">

      <td height="21" colspan="2"><b>Select ActiveRegistry's Location</b></td>
```

```
     <td valign="top" colspan="4" height="21" width="478">
     <select name="arLocation">

<%
             Class.forName("oracle.jdbc.driver.OracleDriver");
             Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
             System.out.println("Database connected");
             Statement statement = dbconn.createStatement();
             ResultSet rst = statement.executeQuery("select arid from ar");
             System.out.println("Query executed");
             String arlist="";

             while(rst.next())
             {
                 arlist=rst.getString("ARID");
%>
      <option value="<%= arlist %>" > <%= arlist %></option>
<%
             } //while
             statement.close();
             dbconn.close();
%>
     </select></td>
</tr>
</table>

<p align="center">
             <input type="submit" name="StopAR" value="Stop AR">
             <input type="reset" name="Submit2" value="Reset Form">
</p>
</form>
</body>
</html>
```

killAR.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>

<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StopAR")!=null)
 {
    String arAddress = (String) request.getParameter("arLocation");
    if(arAddress != null)
    {
      status = (int)Helper.stopAR(arAddress);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   Active Registry stopped successfull
<%
   }
   else
   {
%>
   Invalid parameters: Active Registry not stopped
<%
   }
%>
</center>
</body>
</html>
```

QMStop.jsp

```
<%@ page language="java" import="java.sql.*" %>
<%@ page errorPage="ExceptionHandler.jsp" %>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<table border ="0" cellpadding="0" cellspacing="0">
    <tr>
        <td height="50" valign="middle" bgcolor="#999965">
        <h3><center><b><font size="4">URDS Monitoring and Management System
(URDSMMS)</font></b></center></h3>
        </td>
    </tr>
</table>

<form  name="stopQM" method = "post" action="killQM.jsp" >
<table width="550" border="0" cellspacing="1" cellpadding="0">

<tr>
    <td bgcolor="#CCCCCC" valign="center" height="25" colspan="5" >
                <p align="center">
                <b> Stop QueryManager</b></p>
</tr>
<tr valign="center">

      <td height="21" colspan="2"><b>Select QueryManager's Location</b></td>
      <td valign="top" colspan="4" height="21" width="478">
      <select name="qmLocation">
<%
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection dbconn =
DriverManager.getConnection("jdbc:oracle:thin:@phoenix.cs.iupui.edu:1521:cs9iorcl", "urdsmon", "reddi");
                System.out.println("Database connected");
                Statement statement = dbconn.createStatement();
                ResultSet rst = statement.executeQuery("select qmid from qm");
                System.out.println("Query executed");
                String qmlist="";
                while(rst.next())
                {
                    qmlist=rst.getString("QMID");
%>
        <option value="<%= qmlist %>" > <%= qmlist %></option>
<%
                } //while
                statement.close();
                dbconn.close();
%>

      </select></td>
</tr>
</table>

<p align="center">
                <input type="submit" name="StopQM" value="Stop QM">
                <input type="reset" name="Submit2" value="Reset Form">
</p>

</form>
</body>
</html>
```

killQM.jsp

```
<%@ page import="java.util.*, java.lang.*,RMIHelper"%>
<%@ page errorPage="ExceptionHandler.jsp" %>
<%
 RMIHelper Helper = new RMIHelper();
 int status = -1;
 if(request.getParameter("StopQM")!=null)
 {
    String qmAddress = (String) request.getParameter("qmLocation");
    if(qmAddress != null)
    {
      status = (int)Helper.stopQM(qmAddress);
    }
 }
%>
<html>
<head> <link rel="stylesheet" href="web.css" type="text/css"> </head>
<body bgcolor="#FFFFFF" text="#000000">
<center>
<br>
<br>
<font face="Arial, Helvetica, sans-serif"><b>
<%
   if(status ==0)
   {
     // Display successful mission
%>
   Query Manager stopped successfull
<%
   }
   else
   {
%>
   Invalid parameters: Query Manager not stopped
<%
   }
%>
</center>
</body>
</html>
```